國立政治大學商學院統計學研究所

碩士學位論文

Department of Statistics

College of Commerce

National Chengchi University

Master Thesis

隨機梯度下降法對於順序迴歸模型估計之收斂研究及推薦系統應用

Convergence of Stochastic Gradient Descent for Ordinal Regression Model and Applications for Recommender Systems

研究生: 陳冠廷 撰
Student: Kuan-Ting Chen

指導教授: 翁久幸 博士
Advisor: Ruby Chiu-Hsing Weng, Ph.D.

中華民國 109 年 6 月
June, 2020

DOI:10.6814/NCCU202000780

摘要

矩陣分解是一種普及的協同過濾方法，Koren 和 Sill 在 2011 年提出了基於順序迴歸的矩陣分解方法。相較於傳統的矩陣分解方法，由於基於順序迴歸的矩陣分解方法能夠輸出用戶對物品各項評分的出現機率，因此在應用方面上具有優勢。雖然他們的實驗在準確性上表現優異，但目前尚沒有開源的程式能夠使用。此次論文我們便應用隨機梯度下降法來實現此矩陣分解模型，並討論遭遇到的數值問題，由於此模型涉及順序迴歸模型，我們也研究了順序迴歸模型在隨機梯度下降法下，其參數估計的收斂。

關鍵字:
矩陣分解, 順序迴歸, 隨機梯度下降法, 批次隨機梯度下降法, 平均估計

DOI:10.6814/NCCU202000780

Abstract

Matrix factorization is a popular Collaborating Filtering (CF) method. Koren and Sill (2011) proposed an ordinal regression model with a matrix factorization CF method. This approach is advantageous over traditional matrix factorization methods by its ability to output a full probability distribution of the user-item ratings. Though their experiments showed superior results in its accuracy, there is no publicly available software. In this thesis, we implement the algorithms by Stochastic Gradient Descent (SGD) and discuss the numerical issues encountered. As this approach involves ordinal regression models, we will study the convergence of SGD for ordinal regression models as well.

Keywords:
Matrix Factorization, Ordinal Regression, Stochastic Gradient Descent, Mini-Batch Stochastic Gradient Descent, Average Estimate

DOI:10.6814/NCCU202000780

Contents

DOI:10.6814/NCCU202000780

List of Figures

List of Tables

DOI:10.6814/NCCU202000780

1 INTRODUCTION

With the rapid advancement of technology, more and more industries are put on the market on the Internet, and many people are willing to spend on the Internet because of the convenience brought by the Internet. When a variety of products flood the entire website, although users can have many choices, their consumer experience may be influenced due to the appearance of many products they do not need, which is harmful to businesses. Therefore, if merchants can give users personalized product recommendations, they may improve user satisfaction and loyalty. This personalized recommendation technique is called the recommender system. Many of the top companies like Google, Amazon, Netflix, Spotify have provided their outstanding recommender system.

The mainstream recommender system algorithms can be divided into three types: (1) Collaborative Filtering (CF) recommendation (2) Content-based Recommendation (3) Hybrid recommendation. In this thesis, we focus on the CF algorithm models. CF algorithm is the most widely used algorithm in the field of recommendation. The algorithm does not need to obtain information about users or items in advance. CF relies only on past behavior like product rating or transaction records and model these data to make recommendations to users. And CF algorithm is mainly applied in two models - ***neighborhood approach models*** and *latent factor models*.

Neighborhood approach models recommend users buy the items which the other users with similar interests liked or similar to the items they liked previously. In a sense, the model characterizes users through various items. Latent factor models decompose the user-item rating matrix into the user latent matrix and item latent matrix through matrix factorization and try to explain ratings by these two matrices. Latent factor models in CF area is closely related by ***Singular Value Decomposition***. We included two SVD-like latent factor models in this thesis, "SVD based model" and "SVD++ model".

When the rating is not numerical, it may not be appropriate to directly apply matrix factorization. Although in the common star rating system (For example, when the user rating is between 1 and 5 stars), we can still intuitively treat these ratings as numeric, but if the rating value is S, A, B, C, D, E, it may not be appropriate for us to directly convert them to numerical. Moreover, when the user rates, the distance between the rating values may not be measured at equal intervals. For example, if the user gives a rating A, it may be very close to S, but there is a big gap with B.

To solve the above problem, Koren and Sill proposed OrdRec model [8], an ordinal regression model with a matrix factorization CF method. The ordinal regression model is originally described by McCullagh [9]. OrdRec model treats rating values as ordinal

DOI:10.6814/NCCU202000780

and output a full probability distribution of the ratings through the model. OrdRec takes SVD++ models as the internal model and ordinal regression model as the outer framework. We dub this model "SVD++ based OrdRec".

Though OrdRec in [8] showed superior results in its accuracy, there is no publicly available software. In this thesis, we implement OrdRec by Stochastic Gradient Descent (SGD) and discuss the numerical issues encountered. As this approach involves ordinal regression models, we study the convergence of SGD for ordinal models as well. Although applying SGD to optimize the model parameters is widely used in practice, the convergence properties have not been fully discussed. To compare different types of SGD, we consider SGD, mini-batch SGD, Average SGD (ASGD). ASGD is proposed by Ruppert [12] and Polyak and Juditsky [10], which averages updated parameters in each iteration to estimate model parameters. In this thesis, we found that ASGD outperforms other SGD methods for ordinal regression models. To summarize, this thesis mainly studied the following:

1. Discuss various SGD methods for ordinal regression models

2. Discuss various SGD methods for on OrdRec model with different input sizes

3. Implement two traditional matrix factorization models and two OrdRec models: (1) SVD based model [7] (2) SVD++ model [6] (3) SVD based OrdRec (4) SVD++ based OrdRec [8] and evaluate the performances

DOI:10.6814/NCCU202000780

2 REVIEW

In this section, we review two traditional matrix factorization models and two OrdRec models: (1) SVD based model (2) SVD++ model (3) SVD based OrdRec (4) SVD++ based OrdRec, and related optimized algorithms.

Before beginning, there are some notations to introduce. We assume there are m distinct users, n distinct items, and total N rating records in the dataset. For notational simplicity, we first reindex each users from 1 to m and each items from 1 to n. Then we denote users as u, where $u = 1, 2, 3, ..., m-1, m$, and items as i, where $i = 1, 2, 3, ..., n-1, n$. A rating r_{ui} indicates how much user u like item i after user u used item i, the higher r_{ui} means the more user u like item i. The ratings themselves are ordinal and we assume the rating values range from 1 (extremely dislike) to S (extremely like), that is, $r_{ui} = 1, 2, ..., S-1, S$. Moreover, to distinguish between predicted ratings and actual ratings, we denote predicted ratings by $\widehat{r}_{ui}$. For each user u in the training set, we denote the set of items rated by user u by $R(u)$. The whole training set which containing all rated user-item pairs (u, i, r_{ui}) is denoted by R, and the testing set is denoted by $\widehat{R}$.

2.1 Matrix Factorization Model

Basic matrix factorization model [7, 3] decomposes the user-item rating matrix into two matrices with latent factor space of dimensionality f, one is denoted as user latent matrix P and the other is denoted as item latent matrix Q, and thus user-item interaction can express as inner product of P and Q. Accordingly, user latent matrix P is a $f \times m$ matrix and each user u is associated with a vector $p_u \in \mathbb{R}^f$ in P, and item latent matrix Q is a $f \times n$ matrix and each item i is associated with a vector $q_i \in \mathbb{R}^f$ in Q. For each user u, the element of p_u can be interpreted as the extent of user $u's$ preference for corresponding factors, the element of q_i can be interpreted as the extent to which an item i meets according to the corresponding factors. The result of dot product $q_i^T p_u$ models the interaction between user u and item i which approximates rating r_{ui} rated by user u on item i, that is:

$$\widehat{r}_{ui} = q_i^T p_u \tag{2.1}$$

To learn the latent factor vectors p_u and q_i, the model minimizes the regularized squared error on the set of observed ratings:

$$\min_{p_u,q_i} L \overset{def}{=} \min_{p_u,q_i} \frac{1}{2N} \sum_{(u,i,r_{ui})\in R} (r_{ui} - q_i^T p_u)^2 + \frac{1}{2}\lambda(||p_u||^2 + ||q_i||^2) \tag{2.2}$$

The model learns by fitting previously observed ratings. However, the goal is to predict unrated ratings between users and items. Thus, the model should avoid

DOI:10.6814/NCCU202000780

overfitting the observed data by regularizing the learned parameters, whose magnitudes are penalized. The constant λ controls the extent of regularization and is usually determined by cross-validation.

One of the benefits of applying matrix factorization in CF is the flexibility of adjusting the model based on various views to the data. By extending Equation (2.1) - basic matrix factorization model, it is easy to use the same way to learn the parameters and get more accurate estimated ratings.

SVD based model

Equation (2.1) tried to approximate the ratings by only consider the interactions between users and items. However, much of the observed variations in rating values are due to effects associated with either users or items. These effects are known as biases which are independent of the interactions between users and items. For example, everyone' s rating preferences are different, someone may gives a higher rating to each item while someone may gives a lower rating to each item. Similarly, some popular or perfect items may get higher ratings and some bad items may get lower ratings.

Thus, Koren et al. in [7] modified the basic matrix factorization model by adding latent factors to explain the biases on users and items. The bias form for predicting rating r_{ui} is:

$$b_{ui} = \mu + b_i + b_u$$

The bias form consists of three components: global average μ, user bias for user u, and item bias for item i. By computing the global average of ratings, it can be shown the benchmarks when the users rate the items. The bias term $b_u \in \mathbb{R}^m$ is to explain the difference between user u and other users. The bias term $b_i \in \mathbb{R}^n$ is to explain the difference between item u and other items.

By plugging bias form into original matrix factorization model (Equation (2.1)), new matrix factorization model for predicting r_{ui} is :

$$\widehat{r}_{ui} = b_{ui} + q_i^T p_u = \mu + b_i + b_u + q_i^T p_u \tag{2.3}$$

Now, we dub this new matrix factorization model "SVD based model". SVD based model is able to approximate rating values through different parts: global average, biases for users and items, and interaction between users and items, which is more complete than the original one. Model parameters in SVD based model are optimized by minimizing the regularized squared error function:

$$\min_{p_u, q_i} \frac{1}{2N} \sum_{(u,i,r_{ui}) \in R} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \frac{1}{2}\lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) \tag{2.4}$$

DOI:10.6814/NCCU202000780

By adding bias terms into the original matrix factorization model, more information can be considered and the model can get better performance. Therefore, some improved matrix factorization models are to explain the bias more with various aspects.

SVD++ model

This part reviews SVD++ [6] proposed by Koren. As we mentioned previously, it is free to adjust the model since applying matrix factorization in CF is allowed to various types of input. So far, the model only considers "explicit" feedback. The interest of users on items can be directly obtained through some explicit feedback like user rating history. However, there are many "implicit" feedback deserve attention. For example, everyone can know the user's preference from the purchase evaluation record. However, users go from not yet buying to giving purchase ratings, which are related to many behaviors, such as clicking and dwell time at the product page, which provide a lot of important information about users. Thus, the model can infer user preferences from the more abundant implicit feedback through users' behavior.

There is some implicit feedback can be inferred from observed ratings in the datasets we used in this thesis. These ratings not only show the size of the value but also indicate which items users rate, no matter the users like the items or not. In other words, according to the user's selection and the level of the rating, user preferences can be obtained indirectly.

To integrate implicit and explicit feedback into the model, recall that we denote the set of items rated by user u as $R(u)$, and for each item $j \in R(u)$, we additionally define latent factor $x_j \in \mathbb{R}^f$. Users can be characterized based on these item latent factors which are associated with the items they rated. We dub this model "SVD++" and SVD++ was demonstrated to show superior accuracy by adding the implicit features into model. SVD++ model predicts r_{ui} as follow :

$$\widehat{r}_{ui} = \mu + b_i + b_u + q_i^T(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j) \tag{2.5}$$

By replacing p_u in Equation (2.3) with $p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j$, user matrix P can be described more detailed with the items they rated. Model parameters in SVD++ are optimized by minimizing the regularized squared error function:

$$\min_{p_u, q_i} \frac{1}{2N} \sum_{(u,i,r) \in R} \left(r_{ui} - \mu - b_i - b_u - q_i^T(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j)\right)^2 + \frac{1}{2}\lambda(||p_u||^2 + ||q_i||^2 + ||x_j||^2 + b_u^2 + b_i^2) \tag{2.6}$$

DOI:10.6814/NCCU202000780

2.2 Ordinal Regression Model

Ordinal Regression Model, which is originally described by McCullagh [9], predicts an ordinal value for each sample in the data. For example, the model can predict the performance evaluation level (e.g. A+, A-, ..., F) of the employee in the next quarter based on the past performance in the company.

Given the independent variable $x_i \in \mathbb{R}^f$, assume there is a latent variable $z_i \in \mathbb{R}^1$ has a linear relationship with x_i, that is:

$$z_i = w^T x_i + \xi_i, \quad \xi_i \sim Logistic(0,1) \tag{2.7}$$

where w is the coefficients associated with x_i and $\xi_i \in \mathbb{R}^1$ is the random noise follows logistic distribution $Logistic(0,1)$ with cumulative distribution function $F(x) = 1/(1 + e^{-x})$. Now assume there are S distinct values in the ordinal value set, ordinal regression model predicts r_i, output of the model, by classifying z_i to thresholds of each ordinal value r as follows:

$$r_i = r, \quad if \quad t_{r-1} < z_i \leq t_r, \quad where \quad r = 1, 2, ..., S-1, S \tag{2.8}$$

where $t_r \in \mathbb{R}^1$ are thresholds of each ordinal value and $-\infty = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{S-1} \leq t_S = \infty$. Since these thresholds must be increasing as r grows, the model should be put on some restrictions when directly using $t_1, t_2, ..., t_{S-1}$ as model parameters, this will increase the difficulty when training the model. In [8], Koren and Sill convert these thresholds as follows:

$$t_{r+1} = t_r + e^{\beta_r} \qquad r = 1, 2, ..., S-2 \tag{2.9}$$

By this way, a positive increment in these thresholds can be generate by taking exponential to each β_r. Except t_1 is still a model parameter, $t_2, ..., t_{S-1}$ is no longer the model parameters but β_1, $\beta_2, ..., \beta_{S-2}$.

By Equation (2.7), (2.8) and (2.9), probability of r_i belongs to r can be computed by following calculation:

$$\begin{aligned} P(r_i = r) &= P(t_{r-1} < z_i \leq t_r) \\ &= P(\xi_i \leq t_r - w^T x_i) - P(\xi_i \leq t_{r-1} - w^T x_i) \\ &= F(t_r - w^T x_i) - F(t_{r-1} - w^T x_i) \end{aligned} \tag{2.10}$$

Then probability of r_i less than r is:

$$P(r_i \leq r) = \sum_{s=1}^{r} P(r_i = s) = F(t_r - w^T x_i) = \frac{1}{1 + e^{-(t_r - w^T x_i)}} \tag{2.11}$$

DOI:10.6814/NCCU202000780

In this way, the ordinal regression Model can be applied to fit independent variables x_i with ordinal value r. The model parameter in the ordinal regression model is:

$$\boldsymbol{\theta} = \{w, t_1, \beta_1, \beta_2, ..., \beta_{S-2}\}$$

Model parameters in ordinal regression model are optimized by maximizing regularized log-likelihood function:

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} logP(\widehat{r_i} = r|\boldsymbol{\theta}) - \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \tag{2.12}$$

2.3 OrdRec Model

Koren and Sill proposed a new CF model, OrdRec [8], which takes SVD++ as the internal model and ordinal regression model as the outer framework. Unlike traditional CF models like SVD based model and SVD++ model, OrdRec no longer predicts a value but predicts the probability of belonging to each distinct value, this shows a more flexible option when computing the final output. In this section, we also implement another OrdRec model which takes SVD based model as the internal model and we dub “SVD OrdRec”.

In Section 2.2, ordinal regression model decided predicted value by classifying a latent variable z_i through the threshold of each ordinal value. As long as $w^T x_i$ in Equation (2.7) is changed to the output of the recommender system, ordinal regression model can be applied to the recommender system in the same way. We denote the output of the internal model SVD based model or SVD++ model by y_{ui}, then we can make a latent variable z_{ui}:

$$z_{ui} = y_{ui} + \xi_{ui}, \quad \xi_{ui} \sim Logistic(0,1) \tag{2.13}$$

where ξ_{ui} is the random noise follows logistic distribution $Logistic(0,1)$ with cumulative distribution function $F(x) = 1/(1+e^{-x})$. Similar to Section 2.2, assume there are S distinct values in the ordinal value set, then OrdRec predicts r_{ui} by classifying z_{ui} to thresholds of each ordinal value r as follows:

$$r_{ui} = r, \quad if \quad t_{r-1} < z_{ui} \leq t_r, \quad where \quad r = 1, 2, ..., S-1, S \tag{2.14}$$

where t_r are thresholds of each ordinal value and we convert them by using Equation (2.9). So far, we have introduced all model parameters: parameters included in internal model, t_1, β_1, $\beta_2, ..., \beta_{S-2}$, and we denote them by $\boldsymbol{\theta}$.

By Equation (2.13) and Equation (2.14), we have $P(r_{ui} = r|\boldsymbol{\theta}) = F(t_r - y_{ui}) - F(t_{r-1} - y_{ui})$ and the probability of r_{ui} less than r:

$$P(r_{ui} \leq r|\boldsymbol{\theta}) = F(t_r - y_{ui}) = \frac{1}{1+e^{-(t_r - y_{ui})}} \tag{2.15}$$

DOI:10.6814/NCCU202000780

Model parameters in OrdRec are optimized by maximizing the regularized log-likelihood function in the training set $\boldsymbol{R}$:

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{(u,i,r)\in R} logP(r_{ui} = r|\boldsymbol{\theta}) - \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \tag{2.16}$$

SVD based OrdRec

In this part, SVD based OrdRec is built by taking a SVD based model as an internal model, that is:

$$y_{ui} = b_i + b_u + {q_i}^T p_u \tag{2.17}$$

Compared to Equation (2.3), it is clear that Equation (2.17) is missing one μ. Notice that after y_{ui} was made by the internal model, it will pass the thresholds of the ordinal part and finally output the ordinal rating. For such a constant μ, if it appears in Equation (2.17), there is only a difference in OrdRec which is adding one μ in thresholds. For simplicity, the existence of μ in the ordinal model can be ignored.

In SVD based OrdRec, $\boldsymbol{\theta} = \{b_u, b_i, p_u, q_i, t_1, \beta_1, \beta_2, ..., \beta_{S-2}\}$. Since the datasets (detailed description in the next chapter) used in this thesis only have 5 ordinal rating values, we assume S=5. To learn the model, the goal is to maximize the log-likelihood with regularized term :

$$\max_{p_u,q_i,b_u,b_i,t_1,\beta_1,\beta_2,\beta_3} \frac{1}{N} \sum_{(u,i,r)\in R} logP(r_{ui} = r|\boldsymbol{\theta}) - \frac{\lambda}{2}(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2 + t_1^2 + \beta_1^2 + \beta_2^2 + \beta_3^2) \tag{2.18}$$

SVD++ based OrdRec

In this part, SVD++ based OrdRec is built by taking a SVD++ model as an internal model, that is:

$$y_{ui} = b_i + b_u + q_i^T (p_u + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} x_j) \tag{2.19}$$

Similar to SVD based OrdRec, μ in Equation (2.5) can be ignored when building SVD++ based OrdRec. In SVD++ based OrdRec, $\boldsymbol{\theta} = \left\{b_u, b_i, p_u, q_i, x_j, t_1, \beta_1, \beta_2, ..., \beta_{S-2}\right\}$. Recall the datasets (detailed description in the next chapter) we used in this thesis only have 5 ordinal rating values, we assume S=5. To learn the model, the goal is to maximize the log-likelihood with regularized term :

$$\max_{p_u,q_i,b_u,b_i,x_j,t_1,\beta_1,\beta_2,\beta_3} \frac{1}{N} \sum_{(u,i,r)\in R} logP(r_{ui} = r|\boldsymbol{\theta}) - \frac{\lambda}{2}(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2 + ||x_j||^2 + t_1^2 + \beta_1^2 + \beta_2^2 + \beta_3^2) \tag{2.20}$$

DOI:10.6814/NCCU202000780

2.4 Learning Algorithms

Gradient descent is a fundamental optimization algorithm of a lot of popular algorithms, such as stochastic gradient descent, Adam, etc. Gradient descent is used to minimize objective function by iteratively taking steps proportional (usually called it learning rate γ) to the negative of the gradient (or approximate gradient) of the objective function at the current point. To minimizing an objective function $\boldsymbol{L}$, gradient descent updates $\boldsymbol{\theta}$ in an epoch as follow:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \gamma \nabla L(\boldsymbol{\theta}; R)$$

In gradient descent, a batch is the entire training set $\boldsymbol{R}$. However, when working on a large scale dataset, it often contains billions or even hundreds of billions of examples. Consequently, this large batch may cost a very long time to update parameters once in a single epoch.

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) [11, 5, 1] is widely used in optimizing model parameters. By choosing samples at random from the training set, a big average could estimate from a much smaller one. SGD takes this idea to the extreme – it uses only a single sample $x_i \in R$ (a batch size of 1) per iteration. The term “stochastic” indicates that each sample is randomly chosen. SGD updates $\boldsymbol{\theta}$ in an epoch as follow:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \gamma \nabla L(\boldsymbol{\theta}^{(i-1)}; x_i), i = 1, 2, ..., N \tag{2.21}$$

Given enough epochs, SGD works but is very noisy because it updates parameters by using only one sample in an iteration.

Mini-Batch Stochastic Gradient Descent

Mini-batch stochastic gradient descent (mini-batch SGD) is a compromise method between gradient descent and SGD. The term “mini-batch” indicates that each batch $I_k \subseteq R$ is a subset which is randomly chosen from training samples, where $|I_k| = b \ll N$. Note that each batch is disjoint, if there are $\boldsymbol{K}$ disjoint batches in $\boldsymbol{R}$, it can be shown that $R = I_1 \cup I_2 \cup ... \cup I_K$. Mini-batch SGD updates $\boldsymbol{\theta}$ in an iteration after computing the average gradient value over a subset of the training set, that is:

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \gamma \cdot \frac{1}{b} \sum_{x_i \in I_k} \nabla L(\boldsymbol{\theta}^{(k-1)}; x_i), k = 1, 2, ..., K \tag{2.22}$$

Mini-batch SGD updates parameters more than gradient descent which allows for a more robust convergence, avoiding local minima. Besides, by using a subset as a batch process, model updates provide a more efficient calculation process than SGD.

DOI:10.6814/NCCU202000780

Average Stochastic Gradient Descent

Ruppert [12] and Polyak and Juditsky [10] proposed Average Stochastic Gradient Descent (ASGD) which averages the updated parameters optimized by SGD to estimate $\boldsymbol{\theta}$. Let $L(\boldsymbol{\theta})$ be the objective function of the model, SGD updates the parameter $\boldsymbol{\theta}$ by the following formula:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \gamma_i \nabla L(\boldsymbol{\theta}^{(i-1)}; x_i), \quad i = 1, 2, \dots \tag{2.23}$$

where the learning rate is set as $\gamma_i = \gamma_1 i^{-\alpha}$ with $\gamma_1 > 0$ and $\alpha \in (0.5, 1)$.

We rewrite Equation (2.23) as following:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \gamma_i \varphi(y_i - x_i^T \boldsymbol{\theta}^{(i-1)}) x_i, \quad i = 1, 2, \dots \tag{2.24}$$

The average

$$\overline{\boldsymbol{\theta}}_i = \frac{1}{i} \sum_{t=1}^{i} \boldsymbol{\theta}^{(t)} \tag{2.25}$$

is the final estimator for $\boldsymbol{\theta}$.

Now, consider following assumptions:

Assumption 1:

Let $(\xi_i)_{i\geq 1}$ be a sequence of mutually independent and identically distributed random variables $E(\xi_1) = 0$, $E(\xi_1^2) < \infty$.

Assumption 2:

Let $(x_i)_{i\geq 1}$ be a sequence of mutually independent and identically distributed random variables with dimensionality f, that is, $x_i = \left(x_{i1}, x_{i1}, \dots, x_{if}\right)^T$. $E|x_{ik}|^4 < \infty$ for $1 \leq k \leq f$ and $E(x_1 x_1^T) = B$, $B{>}0$. Sequences $(\xi_i)_{i\geq 1}$ and $(x_i)_{i\geq 1}$ are mutually independent.

Assumption 3:

There exists K_1 such that $|\varphi(u)| \leq K_1(1 + |u|)$ for all $u \in \mathbb{R}^1$.

Given two functions: $\psi(u) = E\varphi(u + \xi_1)$ and $\chi(u) = E\varphi^2(u + \xi_1)$ which are defined under Assumption 1-3. There are some restrictions on ψ and χ.

Assumption 4:

It holds that $\psi(0) = 0$, $u\psi(u) > 0$ for all $u \neq 0$, $\psi(u)$ has a derivative at zero, and $\psi'(0) > 0$. Moreover, there exist $K_2 < \infty$ and $0 < \lambda \leq 1$ such that $|\psi(u) - \psi'(0)u| \leq K_2|u|^{1+\lambda}$.

Assumption 5:

The function $\chi(u)$ is continuous at zero.

Assumption 6:

DOI:10.6814/NCCU202000780

It holds that $(\gamma_i - \gamma_{i+1})/\gamma_i = o(\gamma_i)$, $\gamma_1 > 0$ for all i;

$$\sum_{i=1}^{\infty} \gamma_i^{(1+\lambda)/2} i^{-1/2} < \infty$$

If assumptions 1-6 hold, then there are some convergence properties on $\overline{\boldsymbol{\theta}}_i$. By theorem 4 in [10] proposed by Polyak and Juditsky; that is:

Theorem:

$\overline{\boldsymbol{\theta}}_i \rightarrow \boldsymbol{\theta}$ almost surely and $(\overline{\boldsymbol{\theta}}_i - \boldsymbol{\theta})\sqrt{i}$ converges in distribution to $N(0, V)$, where

$$V = B^{-1} \frac{\chi(0)}{\psi^2(0)}$$

DOI:10.6814/NCCU202000780

3 ORDINAL REGRESSION IMPLEMENTATION

3.1 Numerical Problems in Simulation

When applying Ordinal Regression related models, there will arise some numerical problems when computing the partial derivative of the objective function with respect to each parameter. In this section, we solve the problems by simplifying the equations of the partial derivative of the objective loss function with respect to each parameter.

To compute partial derivative of Equation (2.12) with respect to individual parameter θ_t, we compute partial derivative of $P(r_i \leq r|\boldsymbol{\theta})$ with respect to θ_t first, that is:

$$\begin{aligned}\frac{\partial P(r_i \leq r|\boldsymbol{\theta})}{\partial \theta_t} &= \frac{\partial \frac{1}{1+e^{-(t_r - w^T x_i)}}}{\partial \theta_t} \\ &= \frac{\partial (t_r - w^T x_i)}{\partial \theta_t} P(r_i \leq r|\boldsymbol{\theta})(1 - P(r_i \leq r|\boldsymbol{\theta})) \end{aligned} \tag{3.1}$$

By Equation (2.10), (2.11) and (3.1), we can obtain partial derivative of Equation (2.12) with respect to individual parameter θ_t:

$$\begin{aligned}\frac{\partial L}{\partial \theta_t} &= \frac{1}{N}\sum_{i=1}^{N} \frac{\partial log P(r_i = r|\boldsymbol{\theta})}{\partial \theta_t} - \lambda \theta_t \\ &= \frac{1}{N}\sum_{i=1}^{N} \left(\frac{\frac{\partial (t_r - w^T x_i)}{\partial \theta_t} P(r_i \leq r|\boldsymbol{\theta})(1 - P(r_i \leq r|\boldsymbol{\theta}))}{P(r_i = r|\boldsymbol{\theta})} \right. \\ &\qquad \left. - \frac{\frac{\partial (t_{r-1} - w^T x_i)}{\partial \theta_t} P(r_i \leq r-1|\boldsymbol{\theta})(1 - P(r_i \leq r-1|\boldsymbol{\theta}))}{P(r_i = r|\boldsymbol{\theta})} \right) - \lambda \theta_t \qquad (3.2) \\ &= \frac{1}{N}\sum_{i=1}^{N} \left(\frac{\partial (t_r - w^T x_i)}{\partial \theta_t} \Big(\frac{P(r_i < r|\boldsymbol{\theta}) P(r_i > r|\boldsymbol{\theta})}{P(r_i = r|\boldsymbol{\theta})} + P(r_i > r|\boldsymbol{\theta}) \Big) \right. \\ &\qquad \left. - \frac{\partial (t_{r-1} - w^T x_i)}{\partial \theta_t} \Big(P(r_i < r|\boldsymbol{\theta}) - \frac{P(r_i < r|\boldsymbol{\theta}) P(r_i > r|\boldsymbol{\theta})}{P(r_i = r|\boldsymbol{\theta})} \Big) \right) - \lambda \theta_t \qquad (3.3)\end{aligned}$$

In Equation (3.3), we can find that when computing $\frac{\partial L}{\partial \theta_t}$, the result will be a fraction and the denominator is $P(r_i = r|\boldsymbol{\theta})$. In order to avoid the division error when $P(r_i = r|\boldsymbol{\theta})$ is 0, we simplified the formula and tried to eliminate $P(r_i = r|\boldsymbol{\theta})$ in the denominator.

In Equation (3.3), if $\frac{\partial (t_r - w^T x_i)}{\partial \theta_t} = \frac{\partial (t_{r-1} - w^T x_i)}{\partial \theta_t} = K$, then formula can be simplified to :

$$\begin{aligned}\frac{\partial L}{\partial \theta_t} &= \frac{1}{N}\sum_{i=1}^{N} \left(K \Big(\frac{P(r_i < r|\boldsymbol{\theta}) P(r_i > r|\boldsymbol{\theta})}{P(r_i = r|\boldsymbol{\theta})} + P(r_i > r|\boldsymbol{\theta}) \Big) \right. \\ &\qquad \left. - K \Big(P(r_i < r|\boldsymbol{\theta}) - \frac{P(r_i < r|\boldsymbol{\theta}) P(r_i > r|\boldsymbol{\theta})}{P(r_i = r|\boldsymbol{\theta})} \Big) \right) - \lambda \theta_t \\ &= \frac{1}{N}\sum_{i=1}^{N} K \Big(P(r_i > r|\boldsymbol{\theta}) - P(r_i < r|\boldsymbol{\theta}) \Big) - \lambda \theta_t \qquad (3.4)\end{aligned}$$

DOI:10.6814/NCCU202000780

In ordinal regression model, we have parameters $\boldsymbol{\theta} = \{w, t_1, \beta_1, \beta_2, \beta_3\}$. For parameter w, since $\nabla_w(t_r - w^T x_i) = \nabla_w(t_{r-1} - w^T x_i) = -x_i$, we can directly use Equation (3.4) to compute the partial derivative of Equation (2.12) with respect to w.

Given an independent variable x_i, we have the partial derivative of the objective function L (Equation (2.12)) for w:

$$\nabla L(w) = \frac{1}{N}\sum_{i=1}^{N} -x_i \Big(P(r_i > r|\boldsymbol{\theta}) - P(r_i < r|\boldsymbol{\theta})\Big) - \lambda w$$

For parameters $t_1, \beta_1, \beta_2, \beta_3$, since $\frac{\partial(t_r - w^T x_i)}{\partial\theta_t}$ and $\frac{\partial(t_{r-1} - w^T x_i)}{\partial\theta_t}$ depends on θ_t, we need to compute them base on different ratings by Equation (3.2). The following are partial derivatives of the objective function L (Equation (2.12)) with respect to $t_1, \beta_1, \beta_2, \beta_3$ and we show the detailed computations in Appendix A.

For t_1 :

$$\nabla L(t_1) = \frac{1}{N}\sum_{i=1}^{N} P(r_i > r|\boldsymbol{\theta}) - P(r_i < r|\boldsymbol{\theta}) - \lambda t_1$$

For β_1:

$$\nabla L(\beta_1) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L}{\partial \beta_{1,x_i}} - \lambda\beta_1,$$

$$where \quad \frac{\partial L}{\partial \beta_{1,x_i}} = \begin{cases} 0, \quad if \quad r = 1 \\ \dfrac{(1 + e^{w^T x_i - t_1})(e^{\beta_1 + w^T x_i - t_1 - e^{\beta_1}})}{(e^{w^T x_i - t_1})(1 + e^{w^T x_i - t_1 - e^{\beta_1}})(1 - e^{-e^{\beta_1}})}, \quad if \quad r = 2 \\ e^{\beta_1}(P(r_i > 3) - P(r_i \le 2)), \quad if \quad r = 3 \\ e^{\beta_1}(P(r_i > 4) - P(r_i \le 3)), \quad if \quad r = 4 \\ \dfrac{-e^{\beta_1}P(r_i \le 4)(1 - P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_1}P(r_i \le 4), \quad if \quad r = 5 \end{cases}$$

For β_2:

$$\nabla L(\beta_2) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L}{\partial \beta_{2,x_i}} - \lambda\beta_2,$$

DOI:10.6814/NCCU202000780

$$where \quad \frac{\partial L}{\partial \beta_{2,x_i}} = \begin{cases} 0, \quad if \quad r=1 \\ 0, \quad if \quad r=2 \\ \dfrac{(1+e^{w^Tx_i - t_1 - e^{\beta_1}})(e^{\beta_2 + w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2}})}{(e^{w^Tx_i - t_1 - e^{\beta_1}})(1+e^{w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1-e^{-e^{\beta_2}})}, \quad if \quad r=3 \\ e^{\beta_2}(P(r_i > 4) - P(r_i \le 3)), \quad if \quad r=4 \\ \dfrac{-e^{\beta_2}P(r_i \le 4)(1-P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_2}P(r_i \le 4), \quad if \quad r=5 \end{cases}$$

For β_3:

$$\nabla L(\beta_3) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L}{\partial \beta_{3,x_i}} - \lambda\beta_3,$$

$$where \quad \frac{\partial L}{\partial \beta_{3,x_i}} = \begin{cases} 0, \quad if \quad r=1 \\ 0, \quad if \quad r=2 \\ 0, \quad if \quad r=3 \\ \dfrac{(1+e^{w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2}})(e^{\beta_3 + w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})}{(e^{w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1+e^{w^Tx_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})(1-e^{-e^{\beta_3}})}, \quad if \quad r=4 \\ \dfrac{-e^{\beta_3}P(r_i \le 4)(1-P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_3}P(r_i \le 4), \quad if \quad r=5 \end{cases}$$

Next, we assess the performance of six SGD methods for ordinal regression models. By considering learning algorithms introduced in Section 2.4, the six SGD methods we used here are (1) SGD (2) SGD with γ_n decayed (3) ASGD with γ_n decayed (4) mini-batch SGD (5) mini-batch SGD with γ_n decayed (6) mini-batch ASGD with γ_n decayed. For convenience, we set α as 0.5 in the simulation when applying ASGD to optimize the model.

Let f denote the dimension of independent variables, S denote the number of distinct ratings, and N the number of training samples. We generate data by the following steps:

1. Randomly generate model parameters :
   (a) coefficient vector $w \in \mathbb{R}^f$ from $N(\mathbf{0}, \mathbf{0.1}\boldsymbol{I_f})$
   (b) first threshold $t_1 \in \mathbb{R}$ from $N(0, 1^2)$
   (c) other thresholds $\beta \in \mathbb{R}^{S-2}$ from $N(\mathbf{0}, \boldsymbol{I_{S-2}})$
2. For each x_i and model parameters w, t_1, β, generate the observed rating value :
   (a) Compute w^Tx_i
   (b) Compute $t_{r+1} = t_r + e^{\beta_r}$, $r = 1, 2, ..., S-2$, where $t_0 = -\infty$ and $t_S = \infty$ (See Equation (2.9)).

DOI:10.6814/NCCU202000780

(c) For $r = 1, 2, ..., S-1, S$, compute $P_r = F(t_r - w^T x_i) - F(t_{r-1} - w^T x_i)$ by using $F(x) = 1/(1+e^{-x})$

(d) For each x_i, draw a rating value r as observed rating value from the following multinomial distribution:

$$r \sim multinomial(N = 1, P_1, P_2, ..., P_S)$$

In this simulation, we set training samples N=50,000, dimensionality f=20, number of distinct ratings S=5 and training epoch=1.

By repeating the above steps twenty times, each with different x_i, w, t_1, and β, we obtain twenty synthetic datasets. Table 1 reports the Root Mean Squared Error (RMSE) between the true and estimated parameter values. The reported RMSE is the average over the twenty synthetic datasets.

	w	$\boldsymbol{t_1}$	$\boldsymbol{\beta}$	**Overall**	**Training Time (Second)**
SGD	0.5153	0.5124	0.2375	0.4217	8.6297
SGD with γ_n decayed	0.0700	0.0741	0.0532	0.0657	8.8103
ASGD with γ_n decayed	**0.0342**	**0.0351**	**0.0292**	**0.0328**	8.8103
mini-batch SGD	0.0545	0.0633	0.1658	0.0945	**0.1212**
mini-batch SGD with γ_n decayed	0.2449	0.1609	0.4376	0.2811	0.1302
mini-batch ASGD with γ_n decayed	0.3560	0.2479	0.5877	0.3972	0.1302

Table 1: Simulation results on ordinal regression model with six SGD methods. Results are measured by RMSE

Overall, we can find that the RMSEs between the original parameters and parameters trained by our ordinal regression models are pretty close to 0. This provides strong evidence for the correctness of ordinal regression model implement by myself and proves that the model can give reasonable parameters after training.

When we focus on the performance of various optimization methods, Table 1 shows that using ASGD with γ_n decayed achieved the best performance in six methods. Moreover, if we use methods with γ_n decayed, it is obvious that using SGD is outperform using mini-batch SGD. The reason should be that mini-batch ASGD with γ_n decayed and mini-batch SGD with γ_n decayed are updated after computing loss in a batch, so the number of updates in both methods are less. Due to the learning rate γ_n of these two methods decayed, the update speed is less able to make the loss reach convergence

DOI:10.6814/NCCU202000780

in an epoch.

Although the SGD method can achieve better performance, the time they need to execute an epoch is much greater than the mini-batch methods. So in order to balance performance and execution time, we continued to use original mini-batch SGD in future experiments.

3.2 ASGD for Ordinal Regression Model

In [2, 10], the convergence of linear regression model under Average Stochastic Gradient Descent (ASGD) had been verified by the theorem proposed by Polyak and Juditsky, however, no one has yet verified the convergence of the ordinal regression model under ASGD. In this section, we check the assumptions to determine whether this theorem is also valid for ordinal regression model.

In ordinal regression model, given an independent variable $x_i \in \mathbb{R}^f$, there is a latent variable $z_i \in \mathbb{R}^1$ and coefficients $w \in \mathbb{R}^f$ such that

$$z_i = w^T x_i + \xi_i, \quad \xi_i \sim logistic(0,1) \tag{3.5}$$

where ξ_i is the random noise from logistic distribution. To update the coefficients w, we have update Equation:

$$w_i = w_{i-1} + \gamma_i \cdot -x_i\Big(P(r_i > r) - P(r_i < r)\Big)$$
$$\overline{w}_i = \frac{1}{i}\sum_{t=1}^{i} w_t$$

compare with Equation (2.24), we have:

$$\varphi(z_i - w_{i-1}^T x_i)x_i = x_i\Big(P(r_i > r) - P(r_i < r)\Big) \tag{3.6}$$

Now, we verified whether each assumption holds under the ordinal regression model, if all of the assumptions hold, we would get the theory to support our optimized results by using ASGD under the ordinal regression model. The detailed proofs are presented in Appendix B. First, by Equation (3.6), we have:

$$\begin{aligned}\varphi(z_i - w_{i-1}^T x_i) &= P(r_i > r) - P(r_i < r)\\ &= P(z_i > t_r) - P(z_i \le t_{r-1})\\ &= P(z_i - w_{i-1}^T x_i > t_r - w_{i-1}^T x_i) - P(z_i - w_{i-1}^T x_i \le t_{r-1} - w_{i-1}^T x_i)\end{aligned}$$

And we can convert to

$$\varphi(u) = P(u > t_r - w^T x_i) - P(u \le t_{r-1} - w^T x_i), \quad u \in R^1 \tag{3.7}$$

DOI:10.6814/NCCU202000780

It is easy to show Assumption 1 and Assumption 2 hold by computing basic statistical properties on ξ_i and x_i. For Assumption 3, we know $\varphi(u)$ is the difference of two probability values by Equation (3.7), then we can simply take $K_1 = 1$ to show Assumption 3 holds. For Assumption 6, since the learning rate we chose is equal to Polyak and Juditsky choose in [2] and is verified, that is $\gamma_i = \gamma_1 i^{-\alpha}, \alpha \in (0.5, 1)$, so Assumption 6 must hold. Now, we only need to check Assumption 4 and Assumption 5.

For Assumption 4, we first check $\psi(0) = 0$, by definition of $\psi(u)$, we have:

$$\begin{aligned}\psi(0) &= E\varphi(0 + \xi_1) \\ &= E\big(P(0 + \xi_1 > t_r - w^T x_i) - P(0 + \xi_1 \le t_{r-1} - w^T x_i)\big) \quad \text{(by Equation (3.7))} \\ &= E\big(P(z_i > t_r)\big) - E\big(P(z_i \le t_{r-1})\big) \\ &= 1 - P(r_i \le r) - P(r_i \le r - 1) \end{aligned} \tag{3.8}$$

However, $\psi(0)$ may not be zero since the last two terms in Equation (3.8) represent the probability of r_i is less than $r - 1$ and r respectively. This does not guarantee that the sum of these two terms can be equal to 1.

So far, we know at least one statement in Assumption 4 does not hold (Actually only one statement in Assumption 4 holds), so in theory, based on these assumptions, when we use ASGD to optimize ordinal regression model, there is no way to obtain the same parameter estimation properties as used in the linear model. However, we can still try to use ASGD to optimize ordinal regression model and check the convergence of parameters. If the performance is still good, it means that perhaps we can still prove the convergence properties of the parameters on the theorem as long as we adjust some of the original assumptions.

DOI:10.6814/NCCU202000780

4 RECOMMENDER SYSTEM IMPLEMENTATION

4.1 Optimization Details

In this section, we compute partial derivatives of the objective loss function of each model with respect to corresponding model parameters and apply mini-batch SGD to optimize each model by these partial derivative values.

SVD based model

Given a user u^* and an item i^*, we have the partial derivatives of the objective function L (Equation (2.4)) with respect to p_{u^*}, q_{i^*}, b_{u^*}, b_{i^*}:

- $\nabla L(p_{u^*}) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, u=u^*\}} e_{ui}\cdot q_i + \lambda p_{u^*}$
- $\nabla L(q_{i^*}) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} e_{ui}\cdot p_u + \lambda q_{i^*}$
- $\nabla L(b_{u^*}) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, u=u^*\}} e_{ui} + \lambda b_{u^*}$
- $\nabla L(b_{i^*}) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} e_{ui} + \lambda b_{i^*}$

where we denote prediction error $r_{ui} - \widehat{r}_{ui}$ by e_{ui}, and $\widehat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$

SVD++ model

The partial derivatives of the objective function L (Equation (2.6)) with respect to parameters in SVD++ are the same as that of parameters in SVD based model except for q_i and new parameter x_j. Given a user u^* and an item i^*,

- $\nabla L(q_{i^*}) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} e_{ui}\cdot (p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j) + \lambda q_{i^*}$
- $\forall j \in R(u^*)$:
  $\nabla L(x_j) = -\frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, j\in R(u)\}} e_{ui}\cdot |R(u)|^{-\frac{1}{2}}\cdot q_i + \lambda x_j$

where we denote prediction error $r_{ui} - \widehat{r}_{ui}$ by e_{ui}, and $\widehat{r}_{ui} = \mu + b_i + b_u + q_i^T(p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j$

Recall Section 3.1, we would encounter numeric problems when computing gradient of the objective function, these problems would also happen in SVD based OrdRec and SVD++ based OrdRec. However, we can use the same method to deal with them. According to the same computation method as Equation (3.3), we can obtain partial derivative of Equation (2.16) with respect to individual parameter θ_t:

$$\frac{\partial L}{\partial \theta_t} = \frac{1}{N}\sum_{(u,i,r_{ui})\in R}\Bigg(\frac{\partial(t_r - y_{ui})}{\partial \theta_t}\Big(\frac{P(r_{ui} < r|\boldsymbol{\theta})P(r_{ui} > r|\boldsymbol{\theta})}{P(r_{ui} = r|\boldsymbol{\theta})} + P(r_{ui} > r|\boldsymbol{\theta})\Big)$$

DOI:10.6814/NCCU202000780

$$- \frac{\partial (t_{r-1} - y_{ui})}{\partial \theta_t}\Big(P(r_{ui} < r|\boldsymbol{\theta}) - \frac{P(r_{ui} < r|\boldsymbol{\theta})P(r_{ui} > r|\boldsymbol{\theta})}{P(r_{ui} = r|\boldsymbol{\theta})}\Big)\Big) - \lambda\theta_t \quad (4.1)$$

Similar to Equation (3.4), we can simplify Equation (4.1) if $\frac{\partial(t_r - y_{ui})}{\partial\theta_t} = \frac{\partial(t_{r-1} - y_{ui})}{\partial\theta_t} = K$ and obtain

$$\frac{\partial L}{\partial \theta_t} = \frac{1}{N}\sum_{(u,i,r_{ui})\in R} K\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda\theta_t \quad (4.2)$$

SVD based OrdRec

In Section 2.3, we have parameters $\boldsymbol{\theta} = \{b_u, b_i, p_u, q_i, t_1, \beta_1, \beta_2, \beta_3\}$ in SVD based OrdRec. For parameters $t_1, \beta_1, \beta_2, \beta_3$, the partial derivatives of Equation (2.18) with respect to these parameters are the same as that of parameters in Ordinal regression in Section 3.1, hence we only need to compute the partial derivatives of Equation (2.18) with respect to parameters b_u, b_i, p_u, q_i. Besides, since $\frac{\partial(t_r - y_{ui})}{\partial\theta_t} = \frac{\partial(t_{r-1} - y_{ui})}{\partial\theta_t} = -\frac{\partial y_{ui}}{\partial\theta_t}$, we can directly use Equation (4.2) to compute the partial derivatives of Equation (2.18) with respect to these parameters.

Given a user u^* and an item i^*, we have the partial derivatives of the objective function L (Equation (2.18)) with respect to b_u, b_i, p_u, q_i:

For b_{u^*}:

$$\frac{\partial(t_r - y_{ui})}{\partial b_{u^*}} = \frac{\partial(t_{r-1} - y_{ui})}{\partial b_{u^*}} = -\frac{\partial y_{ui}}{\partial b_{u^*}} = -\frac{\partial(b_{u^*} + b_i + q_i^T p_{u^*})}{\partial b_{u^*}} = -1$$

$$\nabla L(b_{u^*}) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, u=u^*\}} -\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda b_{u^*}$$

For b_{i^*}:

$$\frac{\partial(t_r - y_{ui})}{\partial b_{i^*}} = \frac{\partial(t_{r-1} - y_{ui})}{\partial b_{i^*}} = -\frac{\partial y_{ui}}{\partial b_{i^*}} = -\frac{\partial(b_u + b_{i^*} + q_{i^*}^T p_u)}{\partial b_{i^*}} = -1$$

$$\nabla L(b_{i^*}) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} -\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda b_{i^*}$$

For p_{u^*}:

$$\nabla_{p_{u^*}}(t_r - y_{ui}) = \nabla_{p_{u^*}}(t_{r-1} - y_{ui}) = -\nabla_{p_{u^*}}(y_{ui}) = -\nabla_{p_{u^*}}(b_{u^*} + b_i + q_i^T p_{u^*}) = -q_i$$

$$\nabla L(p_{u^*}) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, u=u^*\}} -q_i\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda p_{u^*}$$

For q_{i^*}:

$$\nabla_{q_{i^*}}(t_r - y_{ui}) = \nabla_{q_{i^*}}(t_{r-1} - y_{ui}) = -\nabla_{q_{i^*}}(y_{ui}) = -\nabla_{q_{i^*}}(b_{u^*} + b_i + q_i^T p_{u^*}) = -p_u$$

$$\nabla L(q_{i^*}) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} -p_u\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda q_{i^*}$$

DOI:10.6814/NCCU202000780

SVD++ based OrdRec

In Section 2.3, we have parameters $\boldsymbol{\theta} = \left\{b_u, b_i, p_u, q_i, x_j, t_1, \beta_1, \beta_2, \beta_3\right\}$ in SVD++ based OrdRec. The partial derivatives of objective function (Equation (2.20)) with respect to SVD++ based OrdRec parameters are the same as that of SVD based OrdRec except for q_i and new parameter x_j.

Given a user u^* and an item i^*, we have the partial derivatives of the objective function L (Equation (2.20)) for parameters q_i and x_j:

For q_{i^*}:

$$\begin{aligned}\nabla_{q_{i^*}}(t_r - y_{ui}) &= \nabla_{q_{i^*}}(t_{r-1} - y_{ui}) = -\nabla_{q_{i^*}}(y_{ui}) = -\nabla_{q_{i^*}}\big(b_u + b_{i^*} + q_{i^*}^T(p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j)\big) \\ &= -(p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j)\end{aligned}$$

$$\begin{aligned}\nabla L(q_{i^*}) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, i=i^*\}} -(p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j)\cdot\Big(&P(r_{ui} > r|\boldsymbol{\theta}) \\ &-P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda q_{i^*}\end{aligned}$$

For x_j, where $j \in R(u^*)$:

$$\begin{aligned}\nabla_{x_j}(t_r - y_{ui}) &= \nabla_{x_j}(t_{r-1} - y_{ui}) = -\nabla_{x_j}(y_{ui}) = -\nabla_{x_j}\big(b_u + b_{i^*} + q_{i^*}^T(p_u + |R(u)|^{-\frac{1}{2}}\sum_{j\in R(u)} x_j)\big) \\ &= -q_i^T|R(u)|^{-\frac{1}{2}}\end{aligned}$$

$$\begin{aligned}\nabla L(x_j) = \frac{1}{N}\sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in R, j\in R(u)\}} -q_i^T|R(u)|^{-\frac{1}{2}}\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) \\ -\lambda x_j\end{aligned}$$

4.2 Recommender System Update Rule

To optimize matrix factorization models, we use mini-batch SGD to balance the efficiency of calculation and the degree of convergence. In mini-batch SGD, we loop over all batches which consist of whole ratings in R and update parameter after computing the partial derivatives of objective function with parameters involved in each batch.

SVD based model

For a given training batch I_k, we modify each parameter in SVD based model by moving in the opposite direction of partial derivative, yielding :

- $\forall u^* \in \{u|(u, i, r_{ui}) \in I_k\}$
  $p_{u^*} \leftarrow p_{u^*} + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui})\in I_k, u=u^*\}} e_{ui} \cdot q_i - \lambda p_{u^*}\right)$

DOI:10.6814/NCCU202000780

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$
  $q_{i^*} \leftarrow q_{i^*} + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} e_{ui} \cdot p_u - \lambda q_{i^*}\right)$

- $\forall u^* \in \{u|(u,i,r_{ui}) \in I_k\}$
  $b_{u^*} \leftarrow b_{u^*} + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, u=u^*\}} e_{ui} - \lambda b_{u^*}\right)$

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$
  $b_{i^*} \leftarrow b_{i^*} + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} e_{ui} - \lambda b_{i^*}\right)$

SVD++ model

For a given training batch I_k, we modify each parameter in SVD++ by moving in the opposite direction of the partial derivative. Recall that the partial derivatives of the objective function with respect to parameters in SVD++ model are as same as that of parameters in SVD based model except for q_i and new parameter x_j, hence the update rule for these parameters are also identical, we only list the update rule for q_i and x_j here.

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$
  $q_{i^*} \leftarrow q_{i^*} + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} e_{ui} \cdot (p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j) - \lambda q_{i^*}\right)$

- $\forall u^* \in \{u|(u,i,r_{ui}) \in I_k\}, \forall j \in R(u^*)$:
  $x_j \leftarrow x_j + \gamma \cdot \left(\frac{1}{b} \cdot \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, j \in R(u)\}} e_{ui} \cdot |R(u)|^{-\frac{1}{2}} \cdot q_i - \lambda x_j\right)$

SVD based OrdRec

For a given training batch I_k, note that our goal is to maximize the objective function, so we will modify parameters of SVD based OrdRec by moving in the same direction of partial derivatives, yielding:

- $\forall u^* \in \{u|(u,i,r_{ui}) \in I_k\}$
  $p_{u^*} \leftarrow p_{u^*} + \gamma \cdot \left(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, u=u^*\}} -q_i \Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda p_{u^*}\right)$

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$
  $q_{i^*} \leftarrow q_{i^*} + \gamma \cdot \left(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} -p_u \Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda q_{i^*}\right)$

- $\forall u^* \in \{u|(u,i,r_{ui}) \in I_k\}$
  $b_{u^*} \leftarrow b_{u^*} + \gamma \cdot \left(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, u=u^*\}} -\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda b_{u^*}\right)$

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$
  $b_{i^*} \leftarrow b_{i^*} + \gamma \cdot \left(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} -\Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta})\Big) - \lambda b_{i^*}\right)$

- $t_1 \leftarrow t_1 + \gamma \cdot \left(\frac{1}{b} \sum_{(u,i,r_{ui}) \in I_k} P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta}) - \lambda t_1\right)$

DOI:10.6814/NCCU202000780

- $\beta_1 \leftarrow \beta_1 + \gamma \cdot (\frac{1}{b}\sum_{(u,i,r)\in R} \frac{\partial L}{\partial \beta_{1,(u,i,r)}} - \lambda\beta_1)$,

  where $\frac{\partial L}{\partial \beta_{1,(u,i,r)}} = \begin{cases} 0, \quad if \quad r=1 \\ \dfrac{(1+e^{y_{ui}-t_1})(e^{\beta_1+y_{ui}-t_1-e^{\beta_1}})}{(e^{y_{ui}-t_1})(1+e^{y_{ui}-t_1-e^{\beta_1}})(1-e^{-e^{\beta_1}})}, \quad if \quad r=2 \\ e^{\beta_1}\Big(P(r_{ui}>3)-P(r_{ui}\le 2)\Big), \quad if \quad r=3 \\ e^{\beta_1}\Big(P(r_{ui}>4)-P(r_{ui}\le 3)\Big), \quad if \quad r=4 \\ \dfrac{-e^{\beta_1}P(r_{ui}\le 4)\Big(1-P(r_{ui}\le 4)\Big)}{P(r_{ui}=5)} = -e^{\beta_1}P(r_{ui}\le 4), \quad if \quad r=5 \end{cases}$

- $\beta_2 \leftarrow \beta_2 + \gamma \cdot (\frac{1}{b}\sum_{(u,i,r)\in R} \frac{\partial L}{\partial \beta_{2,(u,i,r)}} - \lambda\beta_2)$,

  where $\frac{\partial L}{\partial \beta_{2,(u,i,r)}} = \begin{cases} 0, \quad if \quad r=1 \\ 0, \quad if \quad r=2 \\ \dfrac{(1+e^{y_{ui}-t_1-e^{\beta_1}})(e^{\beta_2+y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}})}{(e^{y_{ui}-t_1-e^{\beta_1}})(1+e^{y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}})(1-e^{-e^{\beta_2}})}, \quad if \quad r=3 \\ e^{\beta_2}\Big(P(r_{ui}>4)-P(r_{ui}\le 3)\Big), \quad if \quad r=4 \\ \dfrac{-e^{\beta_2}P(r_{ui}\le 4)\Big(1-P(r_{ui}\le 4)\Big)}{P(r_{ui}=5)} = -e^{\beta_2}P(r_{ui}\le 4), \quad if \quad r=5 \end{cases}$

- $\beta_3 \leftarrow \beta_3 + \gamma \cdot (\frac{1}{b}\sum_{(u,i,r)\in R} \frac{\partial L}{\partial \beta_{3,(u,i,r)}} - \lambda\beta_3)$,

  where $\frac{\partial L}{\partial \beta_{3,(u,i,r)}} = \begin{cases} 0, \quad if \quad r=1 \\ 0, \quad if \quad r=2 \\ 0, \quad if \quad r=3 \\ \dfrac{(1+e^{y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}})(e^{\beta_3+y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}-e^{\beta_3}})}{(e^{y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}})(1+e^{y_{ui}-t_1-e^{\beta_1}-e^{\beta_2}-e^{\beta_3}})(1-e^{-e^{\beta_3}})}, \quad if \quad r=4 \\ \dfrac{-e^{\beta_3}P(r_{ui}\le 4)\Big(1-P(r_{ui}\le 4)\Big)}{P(r_{ui}=5)} = -e^{\beta_3}P(r_{ui}\le 4), \quad if \quad r=5 \end{cases}$

SVD++ based OrdRec

For a given training batch I_k, recall that the partial derivatives of the objective function with respect to parameters in SVD++ based OrdRec model are as same as that of parameters in SVD based OrdRec except for q_i and new parameter x_j, hence the update rule for these parameters are also identical, we only list the update rule for q_i and x_j here.

- $\forall i^* \in \{i|(u,i,r_{ui}) \in I_k\}$

DOI:10.6814/NCCU202000780

$q_{i^*} \leftarrow q_{i^*} + \gamma \cdot \Big(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, i=i^*\}} -(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j) \cdot \Big(P(r_{ui} > r|\boldsymbol{\theta})$
$- P(r_{ui} < r|\boldsymbol{\theta}) \Big) - \lambda q_{i^*} \Big)$

- $\forall u^* \in \{u|(u,i,r_{ui}) \in I_k\}, \forall j \in R(u^*)$:
  $x_j \leftarrow x_j + \gamma \cdot \Big(\frac{1}{b} \sum_{\{(u,i,r_{ui})|(u,i,r_{ui}) \in I_k, j \in R(u)\}} -q_i^T |R(u)|^{-\frac{1}{2}} \Big(P(r_{ui} > r|\boldsymbol{\theta}) - P(r_{ui} < r|\boldsymbol{\theta}) \Big)$
  $- \lambda x_j \Big)$

4.3 Simulation for OrdRec Model

Let f denote the dimension of latent factor, m denote the number of distinct users, n number of distinct items, S denote the number of distinct ratings, and N the number of training samples. We generate data by the following steps:

1. Randomly generate model parameters :
   (a) user latent matrix $P \in \mathbb{R}^{f \times m}$, where each column vector $p_u \sim N(\mathbf{0}, \mathbf{0.1}I_f)$ is latent factor of user u
   (b) item latent matrix $Q \in \mathbb{R}^{f \times n}$, where each column vector $q_i \sim N(\mathbf{0}, \mathbf{0.1}I_f)$ is latent factor of item i
   (c) user bias $b_u \in \mathbb{R}^m$ from $N(\mathbf{0}, I_m)$
   (d) item bias $b_i \in \mathbb{R}^n$ from $N(\mathbf{0}, I_n)$
   (e) first threshold $t_1 \in \mathbb{R}$ from N(0, 1)
   (f) other thresholds $\beta \in \mathbb{R}^{S-2}$ from $N(\mathbf{0}, I_{S-2})$
2. Under all model parameters, generate the observed rating values :
   (a) Compute $y_{ui} = b_i + b_u + {q_i}^T p_u$ (See Equation (2.17)).
   (b) Compute $t_{r+1} = t_r + e^{\beta_r}$, $r = 1, 2, ..., S-2$, where $t_0 = -\infty$ and $t_S = \infty$ (See Equation (2.9)).
   (c) For $r = 1, 2, ..., S-1, S$, compute $P_r = F(t_r - y_{ui}) - F(t_{r-1} - y_{ui})$ by using $F(x) = 1/(1 + e^{-x})$
   (d) For each user u and item i, draw a rating value r as observed rating value from the following multinomial distribution:
   $$r \sim multinomial(N = 1, P_1, P_2, ..., P_S)$$
3. Randomly choose N training samples (u, i, r) from data obtained in 2.

DOI:10.6814/NCCU202000780

To increase the reliability of the model training results, we tried different numbers of users and items, dimension f, and samples N we choose from data, and finally evaluated the difference between original parameters and parameters trained by our ordinal model with RMSE. We divided the simulation into two parts, one is to fit the ordinal model on a smaller number of data and the other is to fit the ordinal model on a larger number of data.

For the first case, we set the number of distinct users m to 100 and the number of distinct items n to 500, to observe the training results of the model with a smaller number of data. Then we set samples N we choose from data to 30,000 and 50,000 respectively.

For the second case, we set the number of distinct users m to 900 and the number of distinct items n to 1600, to observe the training results of the model with a larger number of data. Then we set samples N we choose from data to 50,000 and 80,000 respectively.

For the above two cases, we set dimensionality f to 20, 50, 100 respectively. The reason we choose different N's in both cases is the total ratings depend on the size of the number of distinct users m and the number of distinct items n. For example, the first case we set the number of distinct users m to 100 and number of distinct items n to 500, then the most ratings we can have is $m \times n = 100 \times 500 = 50,000$, so we can not choose N larger than 50,000, we can only choose N based on the size of the number of distinct users m and the number of distinct items n. Results on the two cases are reported in Table 2 and Table 3.

DOI:10.6814/NCCU202000780

RMSE				
N	**Parameters**	**f=20**	**f=50**	**f=100**
30,000	P	0.594	0.534	0.477
	Q	0.296	0.274	0.250
	b_u	0.189	0.171	0.299
	b_i	0.274	0.268	0.308
	t_1	0.147	0.095	0.072
	β	0.061	0.351	0.552
50,000	P	0.710	0.664	0.622
	Q	0.340	0.329	0.309
	b_u	0.216	0.244	0.449
	b_i	0.257	0.250	0.337
	t_1	0.204	0.274	0.093
	β	0.167	0.402	0.721

Table 2: Simulation results when number of distinct users $m = 100$ and number of distinct items $n = 500$. Results measured the RMSE between original parameters and parameters trained by our ordinal model involved in ordinal parameters.

RMSE				
N	**Parameters**	**f=20**	**f=50**	**f=100**
50,000	P	0.229	0.215	0.231
	Q	0.198	0.186	0.198
	b_u	0.273	0.304	0.256
	b_i	0.368	0.397	0.327
	t_1	0.036	0.052	0.292
	β	0.037	0.083	0.246
80,000	P	0.291	0.260	0.285
	Q	0.238	0.218	0.235
	b_u	0.220	0.246	0.246
	b_i	0.290	0.312	0.281
	t_1	0.011	0.114	0.449
	β	0.075	0.154	0.184

Table 3: Simulation results when number of distinct users $m = 900$ and number of distinct items $n = 1600$. Results measured the RMSE between original parameters and parameters trained by our ordinal model involved in ordinal parameters.

DOI:10.6814/NCCU202000780

Compare Table 2 and Table 3 to Table 1, we can find that when we apply the ordinal regression model to the recommender system, in any combination of simulations, the RMSE of each parameter performs worse than the simulations using only the ordinal regression model. The main reason may be that when the ordinal regression model is applied to the recommender system, in addition to the threshold two models in Section 3.1 and Section 4.3 must be estimated, there are two unknown matrices user latent matrix $\boldsymbol{P}$ and item latent matrix $\boldsymbol{Q}$ need to be estimated, and the independent variables X_i in the ordinal regression model is already knowing that it is only necessary to estimate the coefficient vector w, so there will be such a gap in the fit between two models.

When comparing the model fit on a larger number of users and items and a smaller number of users and items separately, we can find that using a larger number of users and items to simulate, the RMSEs of their parameters are slightly better than using a smaller one of users and items. This shows to some extent that the more items and users, the better the model can construct user latent matrix $\boldsymbol{P}$ and item latent matrix $\boldsymbol{Q}$ through the user's rating on the item.

DOI:10.6814/NCCU202000780

5 REAL APPLICATION

5.1 Experiment Dataset

In this experiments, we evaluated 4 models : (1) SVD based model (2) SVD++ model (3) SVD based OrdRec (4) SVD++ based OrdRec on two MovieLens dataset [4] which are MovieLens-100K and MovieLens-1M. The MovieLens datasets were collected over various periods, depending on their size and the rating data consist of user id, movie id, rating value(from 1 to 5), and a timestamp.

MovieLens-100k

MovieLens-100K contains 100,000 anonymous ratings (1-5) from 1,682 movies rated by 943 users. The data was collected by GroupLens Research on the MovieLens web site (movielens.umn.edu) between September 19th, 1997 and April 22nd, 1998. Each User in this data had at least 20 ratings.

MovieLens-1M

MovieLens-1M contain 1,000,209 anonymous ratings (1-5) from 3,952 movies rated by 6,040 MovieLens users who joined MovieLens in 2000. Each user in MovieLens-1M also had at least 20 ratings.

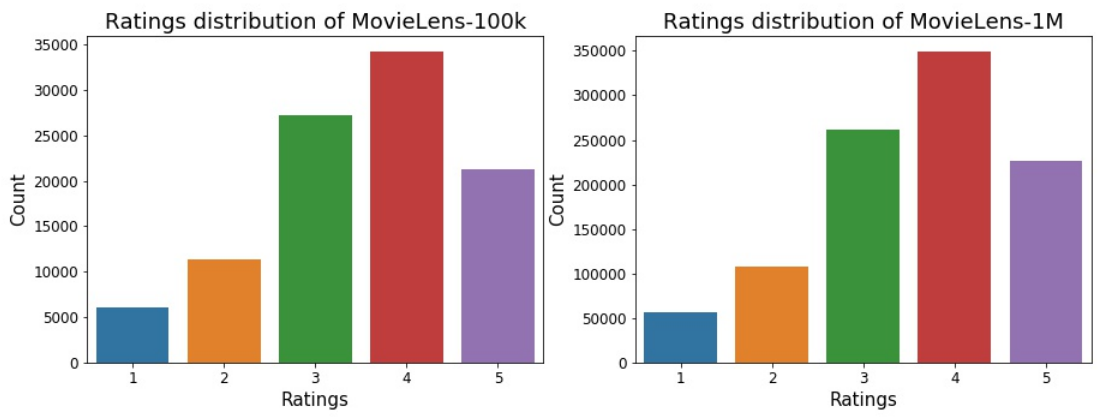


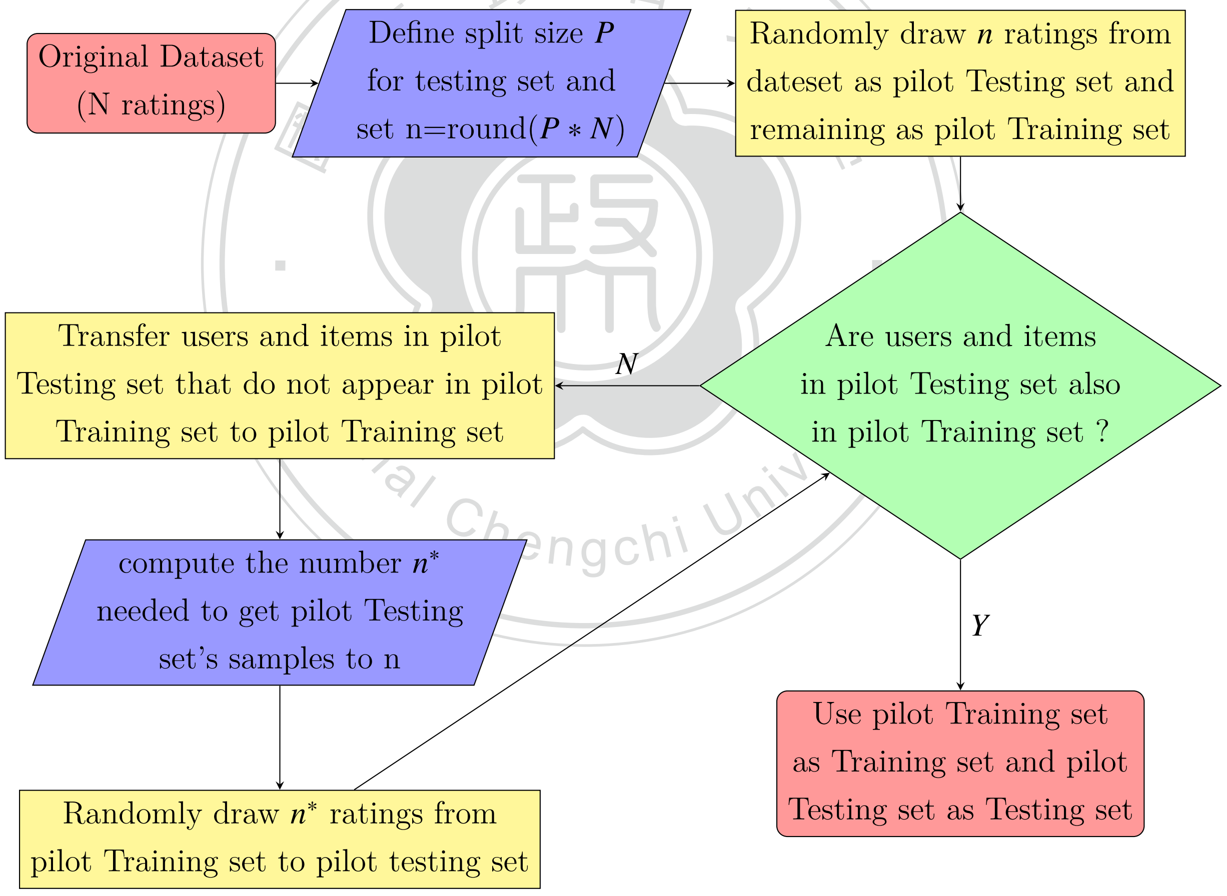
Figure 1: Distributions of ratings of MovieLens-100K and MovieLens-1M

Figure 1 shows the distributions of the ratings of above two datasets, we can obviously find that the user's ratings are almost all concentrated above 3. This may indicate that before watching a movie and giving a rating, the user would first choose a movie that he is more interested in to watch, so after watching the movie, unless the movie is really not good as expected by the user, he would usually give a higher rating.

DOI:10.6814/NCCU202000780

In the matrix factorization model, one of the important problems is "Cold Start", which is the model that can not reference the new user's preferences, nor can it determine the user's preferences for the new item. Personalized recommender systems take advantage of users' past history to make predictions. Providing recommendations to users with no past history becomes a difficult problem for Collaborating Filtering models because their learning and predictive ability are limited. So if we directly randomly split the training set and testing set, we may face "Cold Start" problem.

In practice, there are many ways to solve "Cold Start" problem, like directly estimating the initial value of the model, or averaging user latent matrix and item latent matrix to estimate the latent factor of the new user and new item. No matter which method is used to solve the "Cold Start" problem, as long as the method is fair, we can use this method to evaluate different models. In our experiment, we adjusted the method of splitting the training set and testing set to avoid the user or item in the testing set that did not appear in the training set. Our process follows the flow chart below:


Original Dataset (N ratings)
Define split size P for testing set and set n=round($P * N$)
Randomly draw n ratings from dateset as pilot Testing set and remaining as pilot Training set
Are users and items in pilot Testing set also in pilot Training set ?
N
Transfer users and items in pilot Testing set that do not appear in pilot Training set to pilot Training set
compute the number n^* needed to get pilot Testing set's samples to n
Randomly draw n^* ratings from pilot Training set to pilot testing set
Y
Use pilot Training set as Training set and pilot Testing set as Testing set

According to this data split method, we set the split size P = 0.2 and split the dataset, and we can compute the information as shown in Table 4 below:

DOI:10.6814/NCCU202000780

Dataset	number of users	number of items	|Train|	|Test|
MovieLens-100k	943	1,682	80,000	20,000
MovieLens-1M	6,040	3,952	800,167	200,042

Table 4: Properties of two datasets used in the experiment

5.2 Evaluation Metrics

In our experiment, we use two evaluation metrics, Root Mean Square Error (RMSE) and Fraction Concordant Pairs (FCP), for comparing the performance of different models.

Root Mean Square Error

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far the prediction sample points from the true sample points are. RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is defined as:

$$RMSE(\widehat{R}) = \frac{1}{|\widehat{R}|} \sum_{(u,i,r)\in\widehat{R}} (\widehat{r}_{ui} - r)^2 \tag{5.1}$$

Although RMSE is often used as an indicator of how good a model is, it still has some shortcomings. The value of RMSE is not directly related to the ranking of the item, but this is usually what the recommender system is interested in. A perfectly ranked model may perform poorly on RMSE if the predictions beyond the range of ratings. To monitor the performance of the model in a ranking-wise way, we use another metric —Fraction Concordant Pairs.

Fraction Concordant Pairs

Koren and Sill (2011) proposed Fraction Concordant Pairs (FCP) [8] to give another viewpoint about the result of the CF recommender system. FCP concerns whether the order of the model prediction ratings is consistent with the actual. Given a test set $\widehat{R}$, we compute the number of concordant pairs n_c^u and discordant pairs n_d^u for each user u as follow:

$$n_c^u = |\left\{ (i,j) \mid \widehat{r}_{ui} > \widehat{r}_{uj} \text{ and } r_{ui} > r_{uj} \right\}|,$$
$$n_d^u = |\left\{ (i,j) \mid \widehat{r}_{ui} \geq \widehat{r}_{uj} \text{ and } r_{ui} < r_{uj} \right\}|$$

Let $n_c = \sum_u n_c^u$ and $n_d = \sum_u n_d^u$, FCP can be obtain by computing the proportion of all concordant pairs in all rated item pairs.

DOI:10.6814/NCCU202000780

$$FCP = \frac{n_c}{n_c + n_d} \tag{5.2}$$

5.3 Performance Comparison

We have four matrix factorization models: (1) SVD based model (2) SVD++ model (3) SVD based OrdRec (4) SVD++ based OrdRec, where SVD based model and SVD++ model are aimed at MSE minimization and SVD based OrdRec and SVD++ based OrdRec are aimed at log-likelihood maximization. We evaluated these four models on two datasets, MovieLens-100k and MovieLens-1M, with metrics RMSE and FCP.

In the setting of the learning rate and regularization rate, we set different learning rates and regularization rates for different parameters. Considering that each parameter has a different degree of convergence during the model training process, for example, the gradient of the objective function Equation (2.18) or Equation (2.20) with respect to β_3 is not 0 when the rating value is 4 or more and only updates other than the regularized term in this range, while the gradient of the objective function Equation (2.18) or Equation (2.20) with respect to t_1 is not 0 regardless of the rating, hence there always updates other than regularized term. Therefore, it is better to set different learning rates and regularization rates for each parameter than to set the same one. We determine the epoch, learning rate, and regularization rate according to the change in the loss on the training set. The model stop training when the loss stop increasing or the change is extremely small.

When applying SVD based model and SVD++ model, we set $\gamma = 2.5$ and $\lambda = 0.0004$, and for SVD based OrdRec, we set $\gamma(p_u)$ and $\gamma(q_i)$ as 2.474, $\gamma(b_u)$ and $\gamma(b_u)$ as 2.605, $\gamma(t_1) = 1.816$, $\gamma(\beta_1) = 1.816$, $\gamma(\beta_2) = 1.816$, $\gamma(\beta_3) = 1.684$, $\lambda = 0.0001$, for SVD++ based OrdRec, set $\gamma(p_u)$ and $\gamma(q_i)$ as 2.474, $\gamma(b_u)$ and $\gamma(b_u)$ as 2.605, $x_j = 2.5$, $\gamma(t_1) = 1.816$, $\gamma(\beta_1) = 1.816$, $\gamma(\beta_2) = 1.816$, $\gamma(\beta_3) = 1.684$, $\lambda = 0.0001$. Results on two datasets are reported in Table 5 and Table 6.

DOI:10.6814/NCCU202000780

RMSE			
Method	**f=50**	**f=100**	**f=200**
SVD based model	0.9422	0.9397	0.9375
SVD++ model	**0.9413**	**0.9378**	**0.9365**
SVD based OrdRec	1.2729	1.1859	1.0355
SVD++ based OrdRec	1.0554	1.0469	1.0160
FCP			
Method	**f=50**	**f=100**	**f=200**
SVD based model	0.7128	0.7156	**0.7185**
SVD++ model	**0.7138**	**0.7172**	0.7182
SVD based OrdRec	0.4958	0.5027	0.4765
SVD++ based OrdRec	0.5408	0.5572	0.5442

Table 5: RMSE and FCP on MovieLens-100k testing set under the different models with different dimensionalities. (If RMSE is lower and FCP is higher, the better the model)

RMSE			
Method	**f=50**	**f=100**	**f=200**
SVD based model	0.9149	0.9143	0.9136
SVD++ model	**0.9148**	**0.9142**	**0.9134**
SVD based OrdRec	1.0707	1.0600	1.0461
SVD++ based OrdRec	1.0199	0.9917	0.9901
FCP			
Method	**f=50**	**f=100**	**f=200**
SVD based model	0.7372	**0.7376**	0.7379
SVD++ model	**0.7373**	0.7375	**0.7383**
SVD based OrdRec	0.4317	0.4385	0.4372
SVD++ based OrdRec	0.5618	0.5631	0.5615

Table 6: RMSE and FCP on MovieLens-1M testing set under the different models with different dimensionalities. (If RMSE is lower and FCP is higher, the better the model)

The results show that both baseline models outperform ordinal-based models on both metrics no matter what dataset we use in the experiment. And both models composed of SVD++ model outperform the models composed of SVD based model on both metrics no matter what dataset we use in the experiment. For all models in the experiment, the performance in the two metrics is better as the dimensionality f of latent factor becomes larger.

When RMSE is used as the metric, it is normal for SVD based model and SVD++ model to perform better, because when optimizing the model, these two models minimize MSE, which is directly related to RMSE, that is, to minimize RMSE at the same time, and OrdRec model is optimized to maximize log-likelihood, which is not directly related to RMSE, so there will be a certain degree of weakness in performance.

DOI:10.6814/NCCU202000780

When using FCP as a metric, we are concerned that the model predicts whether the ranking of each user' s rating for each item is consistent with the actual ranking, which is usually the basis for us to determine whether the item recommendation is good or not. This metric is not directly related to any objective loss function of models we used in the experiment, so it is fairer in comparison.

Since the performances in ordinal-based models are much worse than traditional matrix factorization models, we check model convergence in the next few chapters.

Besides, we also compute training time per epoch on two datasets under different models and dimensionalities in Table 7. All models and algorithms in this thesis were implemented in Python and used Numpy and Cython as main packages.

MovieLens-100k			
Method	**f=50**	**f=100**	**f=200**
SVD based model	**0:00:88**	**0:01:05**	**0:01:78**
SVD++ model	0:11:18	0:16:34	0:27:69
SVD based OrdRec	0:01:10	0:01:64	0:02:94
SVD++ based OrdRec	0:12:40	0:18:85	0:33:40
MovieLens-1M			
Method	**f=50**	**f=100**	**f=200**
SVD based model	**0:08:43**	**0:10:11**	**0:16:65**
SVD++ model	3:33:76	5:11:60	15:27:39
SVD based OrdRec	0:11:93	0:18:21	0:37:07
SVD++ based OrdRec	3:53:60	6:27:60	19:34:53

Table 7: Training time (m:s:cs) per epoch on two datasets under different models with different dimensionalities.

5.4 Check for Convergence

To check model convergence, we evaluate the partial derivative value of the objective function with respect to each parameter of each model after training. If the partial derivative value of each parameter in the model is quite close to 0, the model may converge.

Recall that we have computed partial derivatives of the objective loss function of each model with respect to corresponding model parameters. We plug model parameters after training into these partial derivative equations to observe the partial derivative values. For vectors in matrix form like P, Q, b_u, b_i, X (if the model is SVD++), we observed element with the largest and smallest absolute value and mean of all the elements of their partial derivative values; for values t_1, β_1, β_2, β_3, we directly observed their partial derivative values. Results on the two datasets are reported in Table 8 and Table 9.

DOI:10.6814/NCCU202000780

	SVD based model	**SVD++ model**	**SVD based OrdRec**	**SVD++ based OrdRec**
Maximum of $\|\mathbf{P}\|$	0.000126	0.000116	0.000184	0.000172
Minimum of $\|\mathbf{P}\|$	0.000000	0.000000	0.000000	0.000000
Mean of P	0.000000	0.000000	0.000000	0.000000
Maximum of $\|\mathbf{Q}\|$	0.000179	0.000167	0.000359	0.000367
Minimum of $\|\mathbf{Q}\|$	0.000000	0.000000	0.000000	0.000000
Mean of Q	0.000000	0.000000	0.000000	0.000000
Maximum of $\|\mathbf{b_u}\|$	0.000358	0.000362	0.000740	0.000730
Minimum of $\|\mathbf{b_u}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{b_u}$	-0.000008	-0.000007	-0.000049	-0.000012
Maximum of $\|\mathbf{b_i}\|$	0.000419	0.000421	0.000777	0.000778
Minimum of $\|\mathbf{b_i}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{b_i}$	-0.000033	-0.000033	0.000064	0.000085
Maximum of $\|\mathbf{X}\|$		0.003825		0.001223
Minimum of $\|\mathbf{X}\|$		0.000000		0.000000
Mean of X		-0.000029		-0.000005
$\mathbf{t_1}$			0.020466	-0.014648
$\boldsymbol{\beta_1}$			0.024392	-0.017277
$\boldsymbol{\beta_2}$			0.022365	-0.014513
$\boldsymbol{\beta_3}$			0.015285	-0.012566

Table 8: Partial derivative values of the objective function w.r.t each model parameter under each model for MovieLens-100k.

DOI:10.6814/NCCU202000780

	SVD based model	**SVD++ model**	**SVD based OrdRec**	**SVD++ based OrdRec**
Maximum of $\|\mathbf{P}\|$	0.000125	0.000120	0.000178	0.000176
Minimum of $\|\mathbf{P}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{P}$	0.000000	0.000000	0.000000	0.000000
Maximum of $\|\mathbf{Q}\|$	0.000164	0.000164	0.000362	0.000369
Minimum of $\|\mathbf{Q}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{Q}$	0.000000	0.000000	0.000000	0.000000
Maximum of $\|\mathbf{b_u}\|$	0.000773	0.000771	0.001459	0.001455
Minimum of $\|\mathbf{b_u}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{b_u}$	0.000008	0.000008	-0.000068	-0.000062
Maximum of $\|\mathbf{b_i}\|$	0.000689	0.000685	0.001217	0.001235
Minimum of $\|\mathbf{b_i}\|$	0.000000	0.000000	0.000000	0.000000
Mean of $\mathbf{b_i}$	-0.000072	-0.000073	0.000129	0.000138
Maximum of $\|\mathbf{X}\|$		0.022230		0.000859
Minimum of $\|\mathbf{X}\|$		0.000000		0.000000
Mean of $\mathbf{X}$		0.000011		0.000005
$\mathbf{t_1}$			0.038944	0.005847
$\boldsymbol{\beta_1}$			0.045869	0.006383
$\boldsymbol{\beta_2}$			0.049417	0.005693
$\boldsymbol{\beta_3}$			0.015956	0.000025

Table 9: Partial derivative values of the objective function w.r.t each model parameter under each model for MovieLens-1M.

When using different data for training, we can find that no matter which model in the experiment is used, the partial derivative values of each parameter is very close to 0, which means that four models may be trained to convergence.

DOI:10.6814/NCCU202000780

6 CONCLUSION

In this thesis, we have done: (1) Evaluate the convergence of the ordinal regression model under six SGD algorithms. (2) Examine the convergence assumptions of ASGD for ordinal regression model (3) Implement two traditional matrix factorization models and two OrdRec models, and compare the performances.

First, we simulated ordinal regression model and used six SGD algorithms to optimize the model. We found that ASGD with γ_n decayed performed the best. However, some of the assumptions in Polyak and Juditsky [10] do not hold for ASGD. Besides, since it takes a lot of time to execute ASGD with γ_n decayed, we used mini-batch SGD when applying ordinal regression model on the recommender system.

Second, we implement four recommender system models and compare the performances. We found that both SVD based OrdRec and SVD++ based OrdRec performed worse than SVD based model and SVD++ model. These results are different from [8], which may be due to different datasets.

In summary, for OrdRec models, due to the addition of ordinal regression model to the original matrix factorization model, the structure of the model is much more complicated. Some efficient implementations are desirable.

DOI:10.6814/NCCU202000780

Appendix A
Optimization Detail of Ordinal Regression Models

For t_1 :

$$\frac{\partial(t_r - w^T x_i)}{\partial t_1} = \begin{cases} \frac{\partial t_1}{\partial t_1} = 1, \quad if \quad r = 1 \\ \frac{\partial t_2}{\partial t_1} = 1, \quad if \quad r = 2 \\ \frac{\partial t_3}{\partial t_1} = 1, \quad if \quad r = 3 \\ \frac{\partial t_4}{\partial t_1} = 1, \quad if \quad r = 4 \\ \frac{\partial t_5}{\partial t_1} = 0, \quad if \quad r = 5 \end{cases}, \qquad \frac{\partial(t_{r-1} - w^T x_i)}{\partial t_1} = \begin{cases} \frac{\partial t_0}{\partial t_1} = 0, \quad if \quad r = 1 \\ \frac{\partial t_1}{\partial t_1} = 1, \quad if \quad r = 2 \\ \frac{\partial t_2}{\partial t_1} = 1, \quad if \quad r = 3 \\ \frac{\partial t_3}{\partial t_1} = 1, \quad if \quad r = 4 \\ \frac{\partial t_4}{\partial t_1} = 1, \quad if \quad r = 5 \end{cases}$$

$$\frac{\partial L}{\partial t_1} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} \frac{P(r_i \le 1)(1 - P(r_i \le 1))}{P(r_i = 1)} - \lambda t_1 = \\ \qquad \frac{1}{N}\sum_{i=1}^{N} P(r_i > 1) - \lambda t_1, \quad if \quad r = 1 \\ \frac{1}{N}\sum_{i=1}^{N} P(r_i > 2) - P(r_i = 1) - \lambda t_1, \quad if \quad r = 2 \\ \frac{1}{N}\sum_{i=1}^{N} P(r_i > 3) - P(r_i \le 2) - \lambda t_1, \quad if \quad r = 3 \\ \frac{1}{N}\sum_{i=1}^{N} P(r_i > 4) - P(r_i \le 3) - \lambda t_1, \quad if \quad r = 4 \\ \frac{1}{N}\sum_{i=1}^{N} \frac{-P(r_i \le 4)(1 - P(r_i \le 4))}{P(r_i = 5)} - \lambda t_1 \\ \qquad \frac{1}{N}\sum_{i=1}^{N} -P(r_i \le 4) - \lambda t_1, \quad if \quad r = 5 \end{cases}$$

$$= \frac{1}{N}\sum_{i=1}^{N} P(r_i > r|\boldsymbol{\theta}) - P(r_i < r|\boldsymbol{\theta}) - \lambda t_1$$

DOI:10.6814/NCCU202000780

For β_1:

$$\frac{\partial(t_r - w^T x_i)}{\partial\beta_1} = \begin{cases} \frac{\partial t_1}{\partial\beta_1} = 0, \quad if \quad r=1 \\ \frac{\partial t_2}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=2 \\ \frac{\partial t_3}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=3 \\ \frac{\partial t_4}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=4 \\ \frac{\partial t_5}{\partial\beta_1} = 0, \quad if \quad r=5 \end{cases}, \qquad \frac{\partial(t_{r-1} - w^T x_i)}{\partial\beta_1} = \begin{cases} \frac{\partial t_0}{\partial\beta_1} = 0, \quad if \quad r=1 \\ \frac{\partial t_1}{\partial\beta_1} = 0, \quad if \quad r=2 \\ \frac{\partial t_2}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=3 \\ \frac{\partial t_3}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=4 \\ \frac{\partial t_4}{\partial\beta_1} = e^{\beta_1}, \quad if \quad r=5 \end{cases}$$

$$\frac{\partial L}{\partial\beta_1} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial\beta_{1,x_i}} - \lambda\beta_1$$

$$where \quad \frac{\partial L}{\partial\beta_{1,x_i}} = \begin{cases} 0, \quad if \quad r=1 \\ \frac{(1+e^{w^T x_i - t_1})(e^{\beta_1 + w^T x_i - t_1 - e^{\beta_1}})}{(e^{w^T x_i - t_1})(1+e^{w^T x_i - t_1 - e^{\beta_1}})(1-e^{-e^{\beta_1}})}, \quad if \quad r=2 \\ e^{\beta_1}(P(r_i > 3) - P(r_i \le 2)), \quad if \quad r=3 \\ e^{\beta_1}(P(r_i > 4) - P(r_i \le 3)), \quad if \quad r=4 \\ \frac{-e^{\beta_1}P(r_i \le 4)(1-P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_1}P(r_i \le 4), \quad if \quad r=5 \end{cases}$$

when r=2,

$$\begin{aligned} \frac{\partial L}{\partial\beta_1} &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_1}P(r_i \le 2)(1-P(r_i \le 2))}{P(r_i = 2)} - \lambda\beta_1 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_1}(\frac{1}{1+e^{w^T x_i - t_2}})(1-\frac{1}{1+e^{w^T x_i - t_2}})}{\frac{1}{1+e^{w^T x_i - t_2}} - \frac{1}{1+e^{w^T x_i - t_1}}} - \lambda\beta_1 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_1}(\frac{1}{1+e^{w^T x_i - t_2}})(1-\frac{1}{1+e^{w^T x_i - t_2}})}{\frac{e^{w^T x_i - t_1} - e^{w^T x_i - t_1 - e^{\beta_1}}}{(1+e^{w^T x_i - t_1 - e^{\beta_1}})(1+e^{w^T x_i - t_1})}} - \lambda\beta_1 \\ &= \frac{1}{N}\sum_{i=1}^{N} e^{\beta_1}\frac{e^{w^T x_i - t_1 - e^{\beta_1}}}{(1+e^{w^T x_i - t_1 - e^{\beta_1}})^2}\frac{(1+e^{w^T x_i - t_1 - e^{\beta_1}})(1+e^{w^T x_i - t_1})}{e^{w^T x_i - t_1}(1-e^{-e^{\beta_1}})} - \lambda\beta_1 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{(1+e^{w^T x_i - t_1})(e^{\beta_1 + w^T x_i - t_1 - e^{\beta_1}})}{(e^{w^T x_i - t_1})(1+e^{w^T x_i - t_1 - e^{\beta_1}})(1-e^{-e^{\beta_1}})} - \lambda\beta_1 \end{aligned}$$

For β_2:

DOI:10.6814/NCCU202000780

$$\frac{\partial(t_r - w^T x_i)}{\partial\beta_2} = \begin{cases} \frac{\partial t_1}{\partial\beta_2} = 0, \quad if \quad r = 1 \\ \frac{\partial t_2}{\partial\beta_2} = 0, \quad if \quad r = 2 \\ \frac{\partial t_3}{\partial\beta_2} = e^{\beta_2}, \quad if \quad r = 3 \\ \frac{\partial t_4}{\partial\beta_2} = e^{\beta_2}, \quad if \quad r = 4 \\ \frac{\partial t_5}{\partial\beta_2} = 0, \quad if \quad r = 5 \end{cases}, \qquad \frac{\partial(t_{r-1} - w^T x_i)}{\partial\beta_2} = \begin{cases} \frac{\partial t_0}{\partial\beta_2} = 0, \quad if \quad r = 1 \\ \frac{\partial t_1}{\partial\beta_2} = 0, \quad if \quad r = 2 \\ \frac{\partial t_2}{\partial\beta_2} = 0, \quad if \quad r = 3 \\ \frac{\partial t_3}{\partial\beta_2} = e^{\beta_2}, \quad if \quad r = 4 \\ \frac{\partial t_4}{\partial\beta_2} = e^{\beta_2}, \quad if \quad r = 5 \end{cases}$$

$$\frac{\partial L}{\partial\beta_2} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial\beta_{2,x_i}} - \lambda\beta_2$$

$$where \quad \frac{\partial L}{\partial\beta_{2,x_i}} = \begin{cases} 0, \quad if \quad r = 1 \\ 0, \quad if \quad r = 2 \\ \frac{(1 + e^{w^T x_i - t_1 - e^{\beta_1}})(e^{\beta_2 + w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})}{(e^{w^T x_i - t_1 - e^{\beta_1}})(1 + e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1 - e^{-e^{\beta_2}})}, \quad if \quad r = 3 \\ e^{\beta_2}(P(r_i > 4) - P(r_i \le 3)), \quad if \quad r = 4 \\ \frac{-e^{\beta_2}P(r_i \le 4)(1 - P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_2}P(r_i \le 4), \quad if \quad r = 5 \end{cases}$$

when r=3,

$$\begin{aligned} \frac{\partial L}{\partial\beta_2} &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_2}P(r_i \le 3)(1 - P(r_i \le 3))}{P(r_i = 3)} - \lambda\beta_2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_2}(\frac{1}{1+e^{w^T x_i - t_3}})(1 - \frac{1}{1+e^{w^T x_i - t_3}})}{\frac{1}{1+e^{w^T x_i - t_3}} - \frac{1}{1+e^{w^T x_i - t_2}}} - \lambda\beta_2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_2}(\frac{1}{1+e^{w^T x_i - t_3}})(1 - \frac{1}{1+e^{w^T x_i - t_3}})}{\frac{e^{w^T x_i - t_1 - e^{\beta_1}} - e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}}}{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1+e^{w^T x_i - t_1 - e^{\beta_1}})}} - \lambda\beta_2 \\ &= \frac{1}{N}\sum_{i=1}^{N}e^{\beta_2}\frac{e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}}}{(1 + e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})^2}\frac{(1 + e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1 + e^{w^T x_i - t_1 - e^{\beta_1}})}{e^{w^T x_i - t_1 - e^{\beta_1}}(1 - e^{-e^{\beta_2}})} - \lambda\beta_2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{(1 + e^{w^T x_i - t_1 - e^{\beta_1}})(e^{\beta_2 + w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})}{(e^{w^T x_i - t_1 - e^{\beta_1}})(1 + e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1 - e^{-e^{\beta_2}})} - \lambda\beta_2 \end{aligned}$$

For β_3:

DOI:10.6814/NCCU202000780

$$\frac{\partial(t_r - w^T x_i)}{\partial\beta_3} = \begin{cases} \frac{\partial t_1}{\partial\beta_3} = 0, \quad if \quad r=1 \\ \frac{\partial t_2}{\partial\beta_3} = 0, \quad if \quad r=2 \\ \frac{\partial t_3}{\partial\beta_3} = 0, \quad if \quad r=3 \\ \frac{\partial t_4}{\partial\beta_3} = e^{\beta_3}, \quad if \quad r=4 \\ \frac{\partial t_5}{\partial\beta_3} = 0, \quad if \quad r=5 \end{cases}, \qquad \frac{\partial(t_{r-1} - w^T x_i)}{\partial\beta_3} = \begin{cases} \frac{\partial t_0}{\partial\beta_3} = 0, \quad if \quad r=1 \\ \frac{\partial t_1}{\partial\beta_3} = 0, \quad if \quad r=2 \\ \frac{\partial t_2}{\partial\beta_3} = 0, \quad if \quad r=3 \\ \frac{\partial t_3}{\partial\beta_3} = 0, \quad if \quad r=4 \\ \frac{\partial t_4}{\partial\beta_3} = e^{\beta_3}, \quad if \quad r=5 \end{cases}$$

$$\frac{\partial L}{\partial\beta_3} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial\beta_{3,x_i}} - \lambda\beta_3$$

$$where \quad \frac{\partial L}{\partial\beta_{3,x_i}} = \begin{cases} 0, \quad if \quad r=1 \\ 0, \quad if \quad r=2 \\ 0, \quad if \quad r=3 \\ \frac{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(e^{\beta_3 + w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})}{(e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})(1-e^{-e^{\beta_3}})}, \quad if \quad r=4 \\ \frac{-e^{\beta_3}P(r_i \le 4)(1-P(r_i \le 4))}{P(r_i = 5)} = -e^{\beta_3}P(r_i \le 4), \quad if \quad r=5 \end{cases}$$

when r=4,

$$\begin{aligned} \frac{\partial L}{\partial\beta_3} &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_3}P(r_i \le 4)(1-P(r_i \le 4))}{P(r_i = 4)} - \lambda\beta_3 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_3}(\frac{1}{1+e^{w^T x_i - t_4}})(1-\frac{1}{1+e^{w^T x_i - t_4}})}{\frac{1}{1+e^{w^T x_i - t_4}} - \frac{1}{1+e^{w^T x_i - t_3}}} - \lambda\beta_3 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{e^{\beta_3}(\frac{1}{1+e^{w^T x_i - t_4}})(1-\frac{1}{1+e^{w^T x_i - t_4}})}{\frac{e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}} - e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}}}{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})}} - \lambda\beta_3 \\ &= \frac{1}{N}\sum_{i=1}^{N} e^{\beta_3}\frac{e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}}}{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})^2}\frac{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})}{e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}}(1-e^{-e^{\beta_3}})} - \lambda\beta_3 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(e^{\beta_3 + w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})}{(e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2}})(1+e^{w^T x_i - t_1 - e^{\beta_1} - e^{\beta_2} - e^{\beta_3}})(1-e^{-e^{\beta_3}})} - \lambda\beta_3 \end{aligned}$$

DOI:10.6814/NCCU202000780

Appendix B

Assumption Verification

Check of Assumption 1:

Because $\xi_i \sim logistic(0,1)$, we have $E(\xi_1) = 0$ and $var(\xi_1) = \pi^2/3$, then $E(\xi_1^2) = Var(\xi_1) + E(\xi_1)^2 = \pi^2/3 < \infty$

So Assumption 1 holds.

Check of Assumption 2:

In our simulation, we sample $x_i \sim N(\mathbf{0}, I)$, then $E(x_1 x_1^T) = Var(x_1) + E(x_1)E(x_1^T) = I^2 = I$. To compute $E|x_{ik}|^4$ for $1 \le k \le f$, we first consider Moment Generate Function (MGF) of $N(\mathbf{0}, 1)$, that is $M_x(t) = e^{t^2/2}$, then :

$$E|x_{ik}|^4 = E(x_{ik}^4) = M_{x_{ik}}^{(4)}(0) = \frac{d^4 M_{x_{ik}}(t)}{dt^4}\Big|_{t=0} = (t^4 + 6t^2 + 3)e^{t^2/2}\Big|_{t=0} = 3 < \infty$$

So Assumption 2 holds.

Check of Assumption 3:

Because $\varphi(u)$ is the difference of two probability values, we have $|\varphi(u)| \le 1$ and hence we can take $K_1 = 1$ such that $|\varphi(u)| \le 1 \le 1 + |u| = K_1(1 + |u|)$

So Assumption 3 holds.

Check of Assumption 4:

1. Check $\psi(0) = 0$

$$\begin{aligned}
\psi(0) &= E\varphi(0 + \xi_1) \\
&= E\big(P(0 + \xi_1 > t_r - w^T x_i) - P(0 + \xi_1 \le t_{r-1} - w^T x_i)\big) \quad \text{(by Equation (3.7))} \\
&= E\big(P(w^T x_i + \xi_1 > t_r)\big) - E\big(P(w^T x_i + \xi_1 \le t_{r-1})\big) \\
&= E\big(P(z_i > t_r)\big) - E\big(P(z_i \le t_{r-1})\big) \\
&= P(z_i > t_r) - P(z_i \le t_{r-1}) \\
&= 1 - P(z_i \le t_r) - P(z_i \le t_{r-1}) \\
&= 1 - P(r_i \le r) - P(r_i \le r - 1)
\end{aligned} \tag{B.1}$$

may not be zero.

2. Check $u\psi(u) > 0$ for all $u \ne 0$

$$\begin{aligned}
u\psi(u) &= uE\varphi(u + \xi_1) \\
&= uE\big(P(u + \xi_1 > t_r - w^T x_i) - P(u + \xi_1 \le t_{r-1} - w^T x_i)\big) \quad \text{(by Equation (3.7))} \\
&= uE\big(P(\xi_1 > t_r - w^T x_i - u) - P(\xi_1 \le t_{r-1} - w^T x_i - u)\big) \\
&= uE\big(1 - P(\xi_1 \le t_r - w^T x_i - u) - P(\xi_1 \le t_{r-1} - w^T x_i - u)\big) \\
&= uE\big(1 - F_\xi(t_r - w_i^T x_i - u) - F_\xi(t_{r-1} - w_i^T x_i - u)\big)
\end{aligned}$$

DOI:10.6814/NCCU202000780

$$= u\big(1 - F_\xi(t_r - w_i^T x_i - u) - F_\xi(t_{r-1} - w_i^T x_i - u)\big)$$

$$= u(1 - \frac{1}{1 + e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1 + e^{-(t_{r-1} - w_i^T x_i - u)}}) \tag{B.2}$$

may not bigger than zero.

3. Check $\psi(u)$ has a derivative at zero, and $\psi'(0) > 0$

By 2. we have $\psi(u) = 1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}}$, then derivative of $\psi(u)$ to u is:

$$\begin{aligned}
\psi'(u) &= \frac{d\psi(u)}{du} \\
&= \frac{d(1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}})}{du} \\
&= -\frac{d(\frac{1}{1+e^{-(t_r - w_i^T x_i - u)}})}{du} - \frac{d(\frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}})}{du} \\
&= \frac{e^{-(t_r - w_i^T x_i - u)}}{(1 + e^{-(t_r - w_i^T x_i - u)})^2} + \frac{e^{-(t_{r-1} - w_i^T x_i - u)}}{(1 + e^{-(t_{r-1} - w_i^T x_i - u)})^2}
\end{aligned}$$

Then :

$$\begin{aligned}
\psi'(0) &= \frac{e^{-(t_r - w_i^T x_i - 0)}}{(1 + e^{-(t_r - w_i^T x_i - 0)})^2} + \frac{e^{-(t_{r-1} - w_i^T x_i - 0)}}{(1 + e^{-(t_{r-1} - w_i^T x_i - 0)})^2} \\
&= \frac{e^{-(t_r - w_i^T x_i)}}{(1 + e^{-(t_r - w_i^T x_i)})^2} + \frac{e^{-(t_{r-1} - w_i^T x_i)}}{(1 + e^{-(t_{r-1} - w_i^T x_i)})^2} \\
&> 0
\end{aligned}$$

4. Check there exist $K_2 < \infty$ and $0 < \lambda \le 1$ such that $|\psi(u) - \psi'(0)u| \le K_2|u|^{1+\lambda}$:

If $u = 0$, By 2. and 3., we have:

$$\begin{aligned}
|\psi(u) - \psi'(0)u| &= |\psi(0) - \psi'(0)0| \\
&= |\psi(0)| \\
&= |1 - \frac{1}{1 + e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1 + e^{-(t_{r-1} - w_i^T x_i - u)}}|
\end{aligned}$$

By assumption, $|\psi(0) - \psi'(0)0|$ should smaller than $K_2|0|^{1+\lambda} = 0$, but by 1., $1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}}$ may not be zero, so $|1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}}|$ may bigger than zero which contradicts the assumption.

In Assumption 4, all of the conditions except 3. may not be satisfied, hence Assumption does not hold.

Check of Assumption 5:

By definition, we have:

$$\chi(u) = E\varphi^2(u + \xi_1)$$

DOI:10.6814/NCCU202000780

$$
\begin{aligned}
&= E\big(P(u+\xi_1 > t_r - w^T x_i) - P(u+\xi_1 \le t_{r-1} - w^T x_i)\big)^2 \quad \text{(by Equation (3.7))}\\
&= E\big(P(\xi_1 > t_r - w^T x_i - u) - P(\xi_1 \le t_{r-1} - w^T x_i - u)\big)^2\\
&= E\big(1 - P(\xi_1 \le t_r - w^T x_i - u) - P(\xi_1 \le t_{r-1} - w^T x_i - u)\big)^2\\
&= E\big(1 - F_\xi(t_r - w^T x_i - u) - F_\xi(t_{r-1} - w^T x_i - u)\big)^2\\
&= E\big(1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}}\big)^2\\
&= \big(1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - u)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - u)}}\big)^2
\end{aligned} \tag{B.3}
$$

Now, compute $\lim_{u\to 0^+} \chi(u)$, $\lim_{u\to 0^-} \chi(u)$, and $\chi(0)$:

$$
\begin{aligned}
\lim_{u\to 0^+} \chi(u) &= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - 0^+)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - 0^+)}})^2\\
&= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i)}})^2\\
\lim_{u\to 0^-} \chi(u) &= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - 0^-)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - 0^-)}})^2\\
&= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i)}})^2\\
\chi(0) &= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i - 0)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i - 0)}})^2\\
&= (1 - \frac{1}{1+e^{-(t_r - w_i^T x_i)}} - \frac{1}{1+e^{-(t_{r-1} - w_i^T x_i)}})^2
\end{aligned} \tag{B.4}
$$

Since $\lim_{u\to 0^+} \chi(u) = \lim_{u\to 0^-} \chi(u) = \chi(0)$, $\chi(u)$ is continuous at zero. And hence Assumption 5 holds.

Check of Assumption 6:

Learning rate we choose is equal to Polyak and Juditsky choose in [2], that is:

$$\gamma_i = \gamma_1 i^{-\alpha}, \alpha \in (0.5, 1)$$

Then:

$$
\begin{aligned}
\sum_{i=1}^{\infty} \gamma_i^{(1+\lambda)/2} i^{-1/2} &\le \sum_{i=1}^{\infty} \gamma_i^1 i^{-1/2}\\
&= \sum_{i=1}^{\infty} \gamma_1 t^{-\alpha} i^{-1/2}\\
&= \gamma_1 \sum_{i=1}^{\infty} i^{-(\alpha+\frac{1}{2})}
\end{aligned}
$$

By P-series test, $\sum_{i=1}^{\infty} i^{-(\alpha+\frac{1}{2})}$ converges when $\alpha + \frac{1}{2} > 1$, i.e $\alpha > 0.5$, which contains the interval (0.5, 1), then:

$$\sum_{i=1}^{\infty} \gamma_i^{(1+\lambda)/2} i^{-1/2} \le \gamma_1 \sum_{i=1}^{\infty} i^{-(\alpha+\frac{1}{2})} < \infty$$

So Assumption 6 holds.

DOI:10.6814/NCCU202000780

References

[1] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization Methods for Large-Scale Machine Learning. *Siam Review*, 60(2):223–311, 2018.

[2] Yixin Fang, Jinfeng Xu, and Lei Yang. Online Bootstrap Confidence Intervals for the Stochastic Gradient Descent Estimator. *The Journal of Machine Learning Research*, 19(1):3053–3073, 2018.

[3] Simon Funk. Netflix Update: Try This at Home, 2006.

[4] F Maxwell Harper and Joseph A Konstan. The Movielens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):1–19, 2015.

[5] Jack Kiefer and Jacob Wolfowitz. Stochastic Estimation of The Maximum of A Regression Function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[6] Yehuda Koren. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[8] Yehuda Koren and Joe Sill. Ordrec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 117–124, 2011.

[9] Peter McCullagh. Regression Models for Ordinal Data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 42(2):109–127, 1980.

[10] Boris T Polyak and Anatoli B Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[11] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[12] David Ruppert. Efficient Estimations from A Slowly Convergent Robbins-Monro Process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

DOI:10.6814/NCCU202000780